uc3m | Universidad Carlos III de Madrid

TESIS DOCTORAL

Novel Fault Tolerant Multi-Bit Upset (MBU) Error-Detection and Correction (EDAC) architecture

Autor:

ANDRÉS JIMÉNEZ OLAZÁBAL

Director:

DR. JORGE PLEITE GUERRA

DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA

Leganés, Mayo 2018

uc3m | Universidad Carlos III de Madrid

TESIS DOCTORAL

Novel Fault Tolerant Multi-Bit Upset (MBU) Error-Detection and Correction (EDAC) architecture

Autor: ANDRÉS JIMÉNEZ OLAZÁBAL

Director: Dr. JORGE PLEITE GUERRA

Firma del Tribunal Calificador:

Firma

Presidente:

Vocal:

Secretario:

Calificación:

Leganés, de de

Contents

Acknowledgements

First and foremost, I'd like to thank my advisor, **Dr. Jorge Pleite**, who gave me not only support and guidance throughout the thesis, but also complete freedom to pursue the lines I wanted at my pace. Thanks a lot for your patience, understanding, innovative ideas and for the trust you gave me, I can only hope I'll be working hand in hand with people half as you.

Of course, many thanks to **Avelino Martin**, for his support, embarrassing questions and interest shown during development phase.

Special thanks to **my wife and my parents** for their unconditional support and the interest shown during whole phases of this work. This work could not be conducted without their support and patience.

Resumen

Desde el punto de vista de seguridad, la certificación aeronáutica de aplicaciones críticas de vuelo requiere diferentes técnicas que son usadas para prevenir fallos en los equipos electrónicos. Los fallos de tipo hardware debido a la radiación solar que existe a las alturas standard de vuelo, como SEU (Single Event Upset) y MCU (Multiple Bit Upset), provocan un cambio de estado de los bits que soportan la información almacenada en memoria. Estos fallos se producen, por ejemplo, en la memoria de configuración de una FPGA, que es donde se definen todas las funcionalidades. Las técnicas de protección requieren normalmente de redundancias que incrementan el coste, número de componentes, tamaño de la memoria y peso.

En la fase de desarrollo de aplicaciones críticas de vuelo, generalmente se utilizan una serie de estándares o recomendaciones de diseño como ABD100, RTCA DO-160, IEC62395, etc, y diferentes técnicas de protección para evitar fallos del tipo SEU o MCU. Estas técnicas están basadas en procesos tecnológicos específicos como memorias robustas, codificaciones para detección y corrección de errores (EDAC), redundancias software, redundancia modular triple (TMR) o soluciones a nivel sistema.

Esta tesis está enfocada a minimizar e incluso suprimir los efectos de los SEUs y MCUs que particularmente ocurren en la electrónica de avión como consecuencia de la exposición a radiación de partículas no cargadas (como son los neutrones) que se encuentra potenciada a las típicas alturas de vuelo. La criticidad en vuelo que tienen determinados sistemas obligan a que dichos sistemas sean *tolerantes a fallos*, es decir, que garanticen un correcto funcionamiento aún cuando se produzca un fallo en ellos. Es por ello que soluciones como las presentadas en esta tesis tienen interés en el sector industrial.

La Tesis incluye una descripción inicial de la física de la radiación incidente sobre aeronaves, y el análisis de sus efectos en los componentes electrónicos aeronaúticos basados en semiconductor, que desembocan en la generación de SEUs y MCUs. Este análisis permite dimensionar adecuadamente y optimizar los procedimientos de corrección que se propongan posteriormente.

La Tesis propone un sistema de corrección de fallos SEUs y MCUs que permita cumplir la condición de *Sistema Tolerante a Fallos*, a la vez que minimiza los niveles de redundancia y de complejidad de los códigos de corrección. El nivel de redundancia es minimizado con la introducción del concepto propuesto HSB (Hardwired Seed Bits), en la que se reduce la información esencial a unos pocos bits semilla, neutros frente a radiación. Los códigos de corrección requeridos se reducen a la corrección de un único error, gracias al uso del concepto de *Distancia Virtual entre Bits*, a partir del cual será posible corregir múltiples errores simultáneos (MCUs) a partir de códigos simples de corrección.

Un ejemplo de aplicación de la Tesis es la implementación de una *Protección Tolerante a* Fallos sobre la memoria SRAM de una FPGA. Esto significa que queda protegida no sólo la información contenida en la memoria sino que también queda auto-protegida la función de protección misma almacenada en la propia SRAM. De esta forma, el sistema es capaz de auto-regenerarse ante un SEU o incluso un MCU, independientemente de la zona de la SRAM sobre la que impacte la radiación. Adicionalmente, esto se consigue con códigos simples tales como corrección por bit de paridad y Hamming, minimizando la dedicación de recursos de computación hacia tareas de supervisión del sistema.

Abstract

For airborne safety critical applications certification, different techniques are implemented to prevent failures in electronic equipments. The HW failures at flying heights of aircrafts related to solar radiation such as SEU (Single-Event-Upset) and MCU (Multiple Bit Upset), causes bits alterations that corrupt the information at memories. These HW failures cause errors, for example, in the Configuration-Code of an FPGA that defines the functionalities. The protection techniques require classically redundant functionalities that increases the cost, components, memory space and weight.

During the development phase for airborne safety critical applications, different aerospace standards are generally recommended as ABD100, RTCA-DO160, IEC62395, etc, and different techniques are classically used to avoid failures such as SEU or MCU. These techniques are based on specific technology processes, Hardened memories, error detection and correction codes (EDAC), SW redundancy, Triple Modular Redundancy (TMR) or System level solutions.

This Thesis is focussed to minimize, and even to remove, the effects of SEUs and MCUs, that particularly occurs in the airborne electronics as a consequence of its exposition to solar radiation of non-charged particles (for example the neutrons). These non-charged particles are even powered at flying altitudes due to aircraft volume. The safety categorization of different equipments/functionalities requires a design based on fault-tolerant approach that means, the system will continue its normal operation even if a failure occurs. The solution proposed in this Thesis is relevant for the industrial sector because of its Fault-tolerant capability.

Thesis includes an initial description for the physics of the solar radiation that affects into aircrafts, and also the analyses of their effects into the airborne electronics based on semiconductor components that create the SEUs and MCUs. This detailed analysis allows the correct sizing and also the optimization of the procedures used to correct the errors.

This Thesis proposes a system that corrects the SEUs and MCUs allowing the fulfilment of the Fault-Tolerant requirement, reducing the redundancy resources and also the complexity of the correction codes. The redundancy resources are minimized thanks to the introduction of the concept of HSB (Hardwired Seed Bits), in which the essential information is reduced to a few seed bits, neutral to radiation. The correction codes required are reduced to the correction of one error thanks to the use of the concept of interleaving distance between adjacent bits, this allows the simultaneous multiple error correction with simple single error correcting codes.

An example of the application of this Thesis is the implementation of the Fault-tolerant architecture of an SRAM-based FPGA. That means that the information saved in the memory is protected but also the correction functionality is auto protected as well, also saved into SRAM memory. In this way, the system is able to self-regenerate the information lost in case of SEUs or MCUs. This is independent of the SRAM area affected by the radiation. Furthermore, this performance is achieved by means simple error correcting codes, as parity bits or Hamming, that minimize the use of computational resources to this supervision tasks for system.

CHAPTER 1: INTRODUCTION

AEROSPACE INDUSTRY FRAMEWORK

WORK TARGET

ACRONYMS

The aerospace industry's continued improvement of its global competitiveness consists of two main issues that have to be considered: the reduction of economic costs, and safe and reliable technological advances. Both provide additional value to the product developing more flexible, versatile, economical, better and simple systems.

The industry's continued development increasingly demands more complex electronic hardware. This demand is usually expressed in terms of optimizing both performance and cost. However, as Fault Tolerant is a design requirement, design optimization is constrained in critical applications.

For instance, in avionics, safe/reliable and small/light products are targeted. Given that failures in some airborne electronics could cause fatal injuries to passengers, they are designed as a fault tolerant system. The certification process is a guarantee that proper operation will continue in the event of a failure in any of the components.

Avionics systems normally work in a radiated environment that could produce a Single Event Upset (SEU) or even an MCU (Multiple Cell Upset). Several techniques are used nowadays to mitigate memory upsets, [26], [27], [28], [29], [34], [44]. These techniques are based on specific technological processes, such as hardened memories, EDACs, SW redundancy, TMR or system level solutions.

Reconfiguration-based techniques are capable of detecting and correcting errors in an optimized operation. However, they do not meet fault-tolerant requirements because there is no guarantee they will be able to retrieve errors that may occur in the scrubbing function itself.

TMR, [7], can be considered for a feasible solution as a fault-tolerant system, thanks to its high redundancy level. However, there are some applications (such as avionics), in which the increased volume and weight associated with the redundancy level are critical in terms of cost (particularly the weight).

The state-of-the-art of avionics equipment based on complex electronics ruled by aeronautics industry standards [38], [39], [40], [41] and [42], that work in a radiated environment, defines maintenance strategies based on health monitoring systems and redundancies for determining the system status to detect early failure.

In particular, airborne electronics systems are certified under several standards, among them, DO-254 [39] and DO-178C [38]. These standards define structured airborne certification guidance for complex electronics hardware and software-based systems, with statements such as any single failure that could cause a catastrophic event is not allowed. These standards define the Design Assurance Level (DAL) as the design procedure or rigor applied during Design development phase to avoid design errors in system. They are intimately related to the failure conditions of system. The failure probability of each functionality of system is included in the System Safety

Assessment by the aircraft manufacturer and they must comply with the qualitative requirement and DAL.

This work proposes a hybrid optimized redundant EDAC system with a minimum redundancy level for meeting fault tolerant requirements under MCUs, which can be implemented as an additional firmware layer in any programmable memory that is transparent for the programmer.

Figure 1.1 conceptually summarizes the process for attaining the proposed solution. Single Error Correction functions (SEC) are simple and effective, [6] and [7], even though only one error can be retrieved. Their operating performance can be improved if detection is added, serving as an EDAC function, [6] and [7], which only corrects when an actual error is detected.

Facing not only single errors but MCUs is a challenge for the EDAC approach from the practical point of view. BCH code can correct this type of errors, although they are not suitable for practical implementations, [6] and [7]. A solution would be to apply the Interleaving concept, [35], [36], which allows reducing the problem from a multiple error to a single error correction. Interleaving guarantees that no more than one error is included in each scrubbing cycle, enabling MCUs to be retrieved from a set of SEC scrubbings.

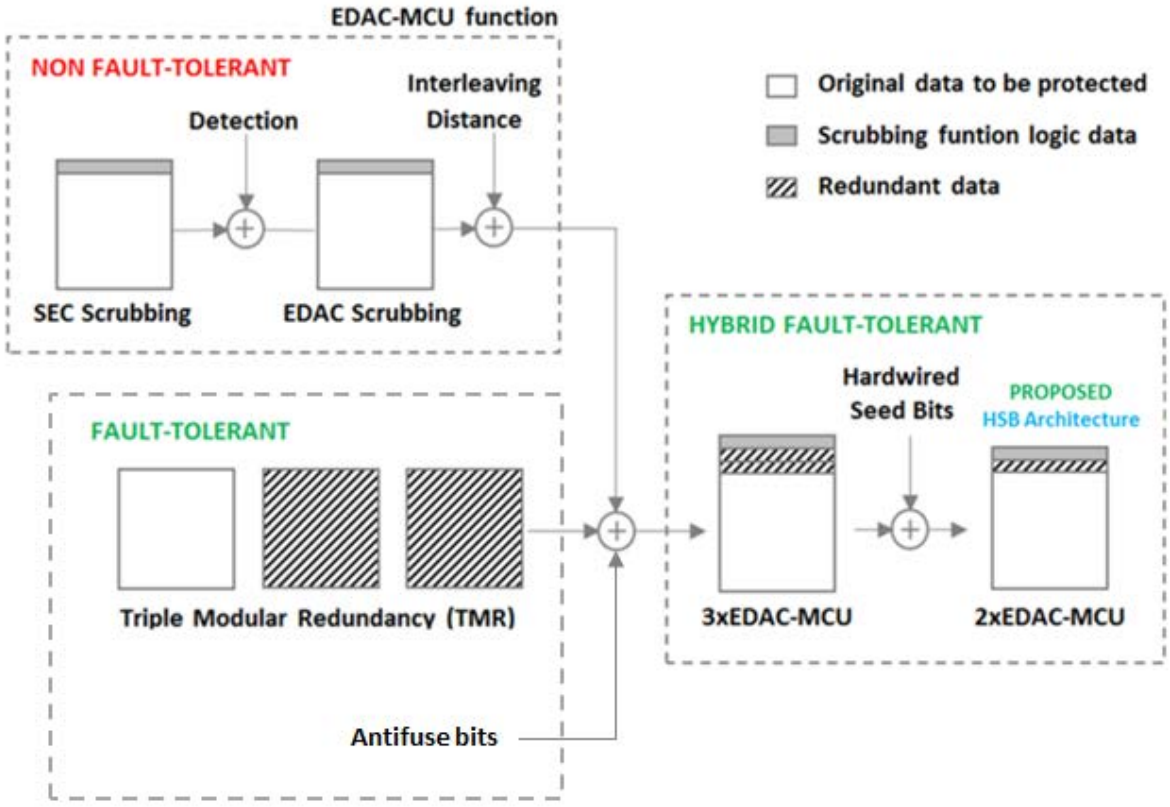


Figure. 1.1. State-of-Art of SEU-MCU protection methods, HSB included.

As a result, the EDAC-MCU approach is very effective, though it has no fault tolerant capabilities. This is because the EDAC function logic itself is exposed to MCUs. On the other hand, TMR is effective for fault tolerant applications, however, it is not efficient due to its redundancy level overhead.

This work discusses the different aspects of both Single and Multiple Event Upset that particularly occurs in airborne electronics. This works proposes that the TMR approach can be hybridized with an EDAC-MCU approach, Fig. 1.1, giving as a result the 3x EDAC-MCU approach, which enables having a fault tolerant system with a low redundancy level, since it only triplicates the EDAC-MCU function. This paper proposes a step forward, which is implementing a 2x EDAC-MCU function while keeping the fault tolerant performance, thanks to HSB, a new concept explained further down.

As a summary to be expanded later, the new approach proposed in this work includes:

- Novel multi-level correction architecture based on a permanent (not vulnerable to radiation) HSB reference that saves the essential information for error retrieval.
- A novel algorithm, which enables autonomous self-detection/correction fault-tolerant capability, with just a 2x EDAC-MCU function approach (called the Control/Monitor in avionics).
- Interleaving concept, [35], [36], which significantly simplifies the correction requirements regarding an MCU, Section IV, reducing the detection/correction action to a SEU approach.
- Detection and correction decoupling, [26], to improve operating time consumption.

1.1 AEROSPACE INDUSTRY FRAMEWORK

The use of this kind of complex electronic hardware devices in the framework of development new products for aerospace applications involves other issues related to nature of these complex hardware electronics devices as ionizing radiation particles. The first failures due to the effects of cosmic rays were observed in electronics in 1975. This kind of radiation has been monitored at A/C flight altitudes since 1992, [25].

Development framework of Aerospace industry is based in a huge number of standards and references that define working procedures. Each of these standards is used for different aspect of design that should be followed by an equipment designer for equipment airborne certification. Some of the most

important ones that define specific requirements to be complied in terms of radiation effects in airborne electronics are:

- **RTCA DO-160** [42], Environmental Conditions and Test Procedures for Airborne Equipment is a standard for environmental test of avionics hardware published by RTCA, Incorporated. This document defines a set of minimal environmental test conditions of the entire spectrum of aircraft and test procedures to be tested in airborne equipments. The purpose of these tests is to define controlled (laboratory) characteristics for assuring the performance of airborne equipment in environmental conditions similar the ones encountered in aircraft operation.

- **ABDX00** [3]**.** Set of Airbus standards related to airborne equipment general design requirements.

- **IEC62395** [2]. Set of standards related to atmospheric radiation effects within avionics electronic equipments. These standards define the ionizing radiation environment that the equipments will be subjected into an aircraft and the potential single event effects (SEE) of the solar radiation environment would have into electronics. These standards also provide a more precise definition of the threat that thermal neutrons pose to avionics as a second mechanism for inducing single event upset (SEU) in microelectronics. The same atmospheric radiation (neutrons and protons) that is responsible for the radiation exposure to avionics electronic equipment is the same that affects to crew and passengers while flying.

- **JESD89A** [1]**,** JEDEC Standard, measurement of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices. It is directly related to soft errors in electronics by atmospheric radiation at altitudes less than 10 000 feet (3.040 m). This standard covers soft errors due to alpha particles, atmospheric neutrons, high-energy neutron and thermal neutron reactions with B^{10}.

- **DO-178C/ EUROCAE ED-12** [38]**.** It defines the Software Certification considerations in Airborne Systems and Equipment Certification. It is a document based on guidance and tasks in terms of safety for software used in airborne systems. It uses a structured guidance and requirements to determine that software performs reliably its intended activities according to the technical specification.

- **DO-254 / EUROCAE ED-80** [39]. It defines the Hardware Certification considerations in Airborne Systems and Equipment Certification. It is a structured guidance for complex electronic hardware design in airborne system in terms of safety requirements. It is intended for Complex electronic hardware devices as Field Programmable Gate Arrays (FPGAs), Programmable Logic Devices (PLDs), and Application Specific Integrated Circuits (ASICs).

- System safety assessment processes as **SAE ARP-4761** [40] and Aircraft development process as **SAE ARP-4754A** [41] are used at aircraft system level to define the safety objectives applicable to the functionalities of a system design determining the criticism or the severity of each function of system. Atmospheric radiation effects are one factor that contributes to the system failure rates.

Therefore, for an aerospace development framework, these standards shall be followed obtaining some general guidance on atmospheric radiation effects on airborne electronics working up to 40.000 feet (12 km). These standards define the ionizing radiation environment that will occur in airborne equipments. They also provide design considerations to calculate the contributions of those effects within avionics systems.

Furthermore, these standards also define the Single Event Effects (SEE) of this cosmic radiation in electronic complex devices. As commented before, the first failure in complex electronics due to cosmic rays (ionizing radiation particles) was observed in 1975 and it is currently monitored in commercial fleets since 1992. This kind of failure is an Hardware issue that occurs when a bit is upset due to the radiation effects of particles as protons, neutrons, alpha particles or heavy ions that are present in airborne environment and produced interactions with hydrogenous materials within aircraft.

1.2 WORK TARGET

Within the aerospace work frame, different mitigation approaches are classically included in the aerospace electronic systems in order to mitigate the effects of the SEU and/or MCU. All the potential SEU/MCU mitigation techniques require additional HW or SW added to the system.

The solution intended in this Thesis is based on an error correcting technique with a novel permanent reference not affected by SEU/MCU, called “Hardwired Seed Bits”, HSB, physically soldered on the PCB. From this permanent reference, the retrieval of original code could be done with a reconstruction as soon as possible of the original values saved in the memory after SEE event.

Main aim of the solution proposed is the MCU protection. As MCU appears in bits physically near each other, single-error correcting techniques fails in the detection of MCU events. Thus, the usual approach is to use a Multiple-error correcting codes, but resources required increase exponentially due to coding complexity. The proposal is to use an algorithm (interleaving distance) that suitably virtually separates the bits into groups to guarantee no more than one upset per group when correcting, even under an MCU event.

Final goal of this work is also the application of this kind of strategy for MCU correction into fault tolerant systems. These kinds of systems are normally demanded into aerospace applications whose operating failure could cause catastrophic effects into the aircraft, crew and passengers. Fault tolerant feature is typically achieved by the triplication of systems, whereas this work allows achieving same protection as triplication fault tolerant systems with just a simple duplication, thanks to the self-detection capability offered by HSB.

These weighted error correction procedures requires extra timing to proceed, based on a complex codification, to protect the memory. An optimization in terms of timing performances is also requested by simplifying the detection and correction procedures. Since correction process takes much longer computation timing than correction and most of times there is no error induced in memory under a normal (low) SEU rate, decoupling both processes would achieve a timing optimization, because correction could be only launched when an error is previously detected.

1.3 ACRONYMS

BCH: Bose, Chaudhuri and Hocquenghem correction code
BPSG: Boron Phosphor Silicate Glass
CEH: Complex Electronic Hardware
COTS:Commercial Off The Shelf hardware component
CPLD: Complex programmable logic device
DAL: Design Assurance Level
DMR: Double Modular Redundancy
EDAC: Error Detection And Correction code
EHC: Electronic Hardware Component
ExT: Exposition Time
FFPA: Functional Failure Path Analysis
FPGA: Field programmable gate array
GCR: Galactic Cosmic Rays
HDL: Hardware Description Languages
HSB: Hardware Seed Bits
IC: Integrate Circuit
ID: Interleaving distance in Frame architecture
IP: Impact probability
IR: Impact rate
LET: Linear Energy Transfer
LRU: Line replaceable unit
MBU: Multiple Bit Upset
MCU: Multiple Cell Upset
MEC: Multiple Error Correction code
MTBU: Mean Time Between Upset
n-MU-IP: n-Multiple Uncorrectable Impact Probability
n-MU-IR: n-Multiple Uncorrectable Impact Rate
OH: Operating Hours in fligh
RTL: Register Transfer Level
SC-IP: Single Correctable Impact Probability
SC-IR: Single Correctable Impact Rate
SU-IP: Single Uncorrectable Impact Probability
SU-IR: Single Uncorrectable Impact Rate
SEC: Single Error Correction Code
SEE: Single Event Effect
SEH: Simple Electronic Hardware
SEU: Single Event Upset
TFP: Total Failure Probability
TFR: Total Failure Rate
TMR: Triple Modular Redundancy

CHAPTER 2: STATE OF ART

AIRBORNE ELECTRONICS NEUTRON RADIATION:
PHYSICS OF THE PROBLEM

SINGLE EVENT UPSET (SEU)

MULTIPLE BIT UPSET (MCU)

CURRENT SOLUTIONS IN AIRBORNE ELECTRONICS

For the last 280 years, the sun activity cycle has been fixed around 11 years. The solar cycle is divided into two phases, solar minimum during 4.8 years and solar maximum about the rest 6.2 years. In these periods, energetic particles are emitted and, as interactions of Galactic Cosmic Rays (GCR), the effects are attenuated by the Earth's magnetosphere.

2.1 AIRBORNE ELECTRONICS NEUTRON RADIATION: PHYSICS OF THE PROBLEM

Energetic particles are emitted as interactions of Galactic Cosmic Rays (GCR). These interactions against the upper atmosphere create a flux build-up of charged particles and neutrons at the altitude level called “Pfotzer maximum”, 60.000 feets, that is not uniform around Earth (depends on altitude, latitude and magnetic field of Earth). In general, for avionics application purposes, a conservative approach is to use an estimated total flux around 9.200 n/cm^2/h at 40.000 feets and 45º latitude. It could have a peak up to 12.000 n/cm^2/h that depends on altitude and Rigidity cut-off. These data commonly agree with different standards such as JEDEC [1], IEC62396 [2] and ABD100 [3]. This flux is approximately 300 times lower at ground level. Above “Pfotzer maximum” altitude, there is a significant abundance of cosmic ray heavy ions.

When heavy ions pass through the silicon substrate of an electronic device [2] and [4], they deposit charge in matter (LET: Linear Energy Transfer, energy deposited per unit path length in a semiconductor along the path of the radiation expressed in MeV·cm^2/mg.) along its range (μm) by direct ionization. When non-charged particles as neutrons pass through the silicon substrate of a device, it cannot deposit charge in matter, but interacts with matter causing indirect ionization. These indirect ionization interactions are neutron-nuclei reactions based mainly on absorption and scattering.

When absorption of neutron occurs in indirect ionization [4], the excited nucleus breaks up into several lighter nuclei that deposits charge in matter (LET) along its range (μm) by direct ionization. LET is dependent on the mass, energy of the particle and the matter which it travels in. As a general rule linked to momentum and energy conservation laws, lighter particles have lower LET and higher range than heavier ones, emitted close to their Bragg peak [4],[19]. Lighter particles must travel some distance to reach their Bragg peak, occurring immediately before the particle come to rest. These effects have been calculated by means Ziegler’s range tables, where LET deposition in Silicon matter of different lighter particles as He and H are represented. In case of alpha particle, α, Bragg peak occurs because particle deposits more dose along its path in matter as kinetic energy decreases [4], [19]. Alpha particle captures two electrons being thus converted into a neutral atom no longer capable of producing ionization.

LET deposited along ion range creates by direct ionization a track of electrical interactions that could be enough to remove the orbital electrons from atoms of matter, leaving behind a number of electron-hole pairs as it move on in

its track [4], [18], and [19]. A charge event occurs within a nanosecond. Electrons are grouped instantaneously by the electric field near to the final region of the track of ion range. This creates a transient of current that depends on the number of electron-hole pairs affected within the ion track [18]. In this scenario, if the transient charge is higher than critical value of the volume of bit, it could produce an upset. In this way, the information stored in this memory bit is corrupted.

Scattering interactions of mainly hi-energy neutrons outside the aircraft appear mainly because of the presence of hydrogenous materials inside aircraft [21]. Hydrogen is, by far, the main cause of the energy reduction of the fast neutrons coming from Cosmic Rays. These interactions are neutron-nuclei reactions based on scattering, where nucleus interactions reduces the neutron kinetic energy. As heavier the nucleus is, less energy is removed from neutron in each interaction. After a sufficient number of these interactions against hydrogenous materials, the kinetic energy of neutrons is reduced about seven orders of magnitude as explained later in Figure 2.5. The limit of interactions is defined by the molecules of the aircraft volume tempered at 20ºC that corresponds to a 0.025eV (neutrons at energies lower than 1eV are considered as Epithermal and Thermal neutrons).

Shielding calculations have been performed to analyze neutron fluxes inside aircraft as represented in Figure 2.1.a and Figure 2.1.b. A medium-range twin-engine transport plane has been simulated. Internal volume of this transport plane has been homogeneous distributed with humans, plastics, air, fuel and metal alloy.

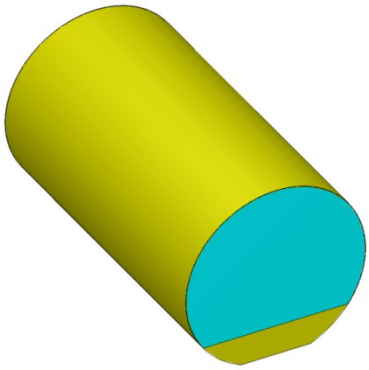


Figure 2.1a
Cargo Bay model of a medium-range twin-engine transport plane

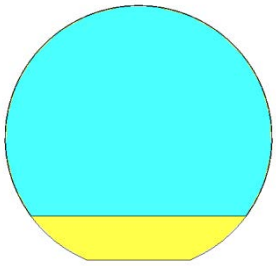


Figure 2.1b
Front view of Cargo Bay model of a medium-range twin-engine transport plane

By means of these scattering interactions calculated, neutron energy spectrum varies inside aircraft. They reduce the proportion of hi-energy

neutrons by increasing the neutrons at lower energies. Thermal neutron (<1 eV) fraction increases while hi-energy neutrons are reduced (>1 MeV) inside aircraft volume, [21].

Model used in shielding calculation is a simplification of the real architecture of fuselage of the cargo bay composed by aluminum alloy based on (2024) whose thickness has been adjusted in such way as the outside neutron flux of aircraft cross over an equivalent mass of such aluminum alloy. Volume inside of aircraft fuselage models an homogeneous material mixture of humans, plastics and other materials based on (H, C, O) that have special interest in the scattering interactions of hi-energy neutrons (neutrons at kinetic energies typically between hundreds of KeV and tens of MeV). Scattering property of these materials comes from its content of light materials able to yield neutrons when nucleus collision occurs. Due to H_2 nucleus have similar mass as neutrons, they are the ones with higher scattering capabilities. As mass increases in components as C, O, Al, scattering capabilities are reduced because of kinetic collision. Incident neutrons will lose less energy in scattering interactions in every collision. Table 2.1 shows the composition used in simulations including the materials located inside aircraft volume. Homogeneous material placed inside aircraft volume is composed by a 71,29% of steel, 14,26% of plastics, 14,26% of humans and a 0.002% of air.

Element	Al alloy	Humans	Steel	Plastics	Air
Al	94.98				
Li	2.2				
Mg	0.12				
Cu	2.7				
H		10.15		4.84	
O		64.46			21
N		3.14		56.72	79
C		18.23	0.5	38.44	
Ca		1.99			
P		1.09			
Cl		0.17			
K		0.38			
S		0.26			
Na		0.11			
Mg		0.04			
Fe			99.5		

Table 2.1. Material distribution inside aircraft volume.

The source of atmospheric neutrons used in shielding calculations have been obtained from NASA flight data at 40.000 fts (12 km approx.) and 45º latitude [2]. These atmospheric neutron fluxes from outside of aircraft have been used, Figure 2.2, in calculations to obtain the atmospheric neutron fluxes inside

aircraft once crossed over the fuselage and interacts with the homogenous volume inside aircraft.

Hi-energy neutron range used from Figure 2.2 is only the ones in the range of energies between 1 to 20 MeV. In order not to disturb the results obtained in simulation, no other range of energies has been used as source in calculation. Therefore, results obtained in the complete energy distribution are exclusively obtained from the scattering interactions from this range of neutron energy, 1 to 20 MeV.

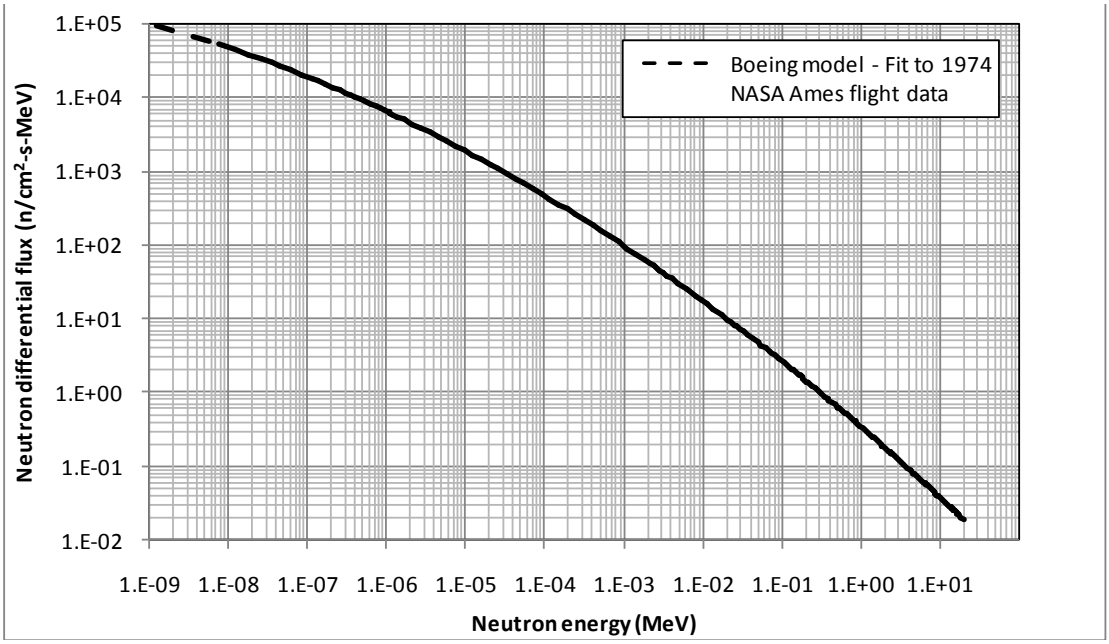


Figure 2.2. Energy spectral distribution of neutron flux at 40.000 fts outside of aircraft.

Figure 2.3 represents the source of neutrons outside of aircraft used for calculations (NASA flight data) and also the flux inside aircraft obtained from the simulations at same range of energies, 1 to 20Mev. As shown in Figure 2.3 the flux inside aircraft is clearly reduced due to scattering interactions with the homogeneous medium inside aircraft volume. The flux is reduced more than one order of magnitude according to calculations performed.

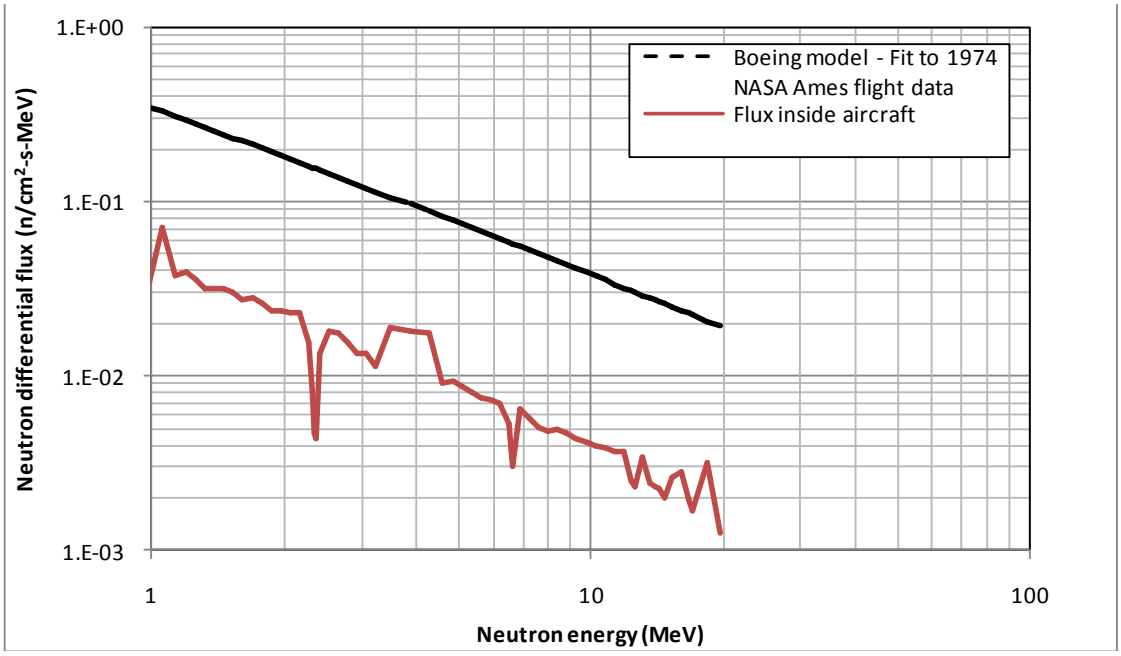


Figure 2.3. Aircraft neutron energy spectral distribution at 40.000 fts

As Figure 2.4 shows, elastic scattering is the most probabilistic reaction. In this reaction the neutron loses energy in the collision against the proton of Hydrogen nucleus. In the range of high energies, the probability of any other reaction is almost negligible being the radioactive capture (n, g) in range of 4 orders of magnitude lower than elastic scattering.

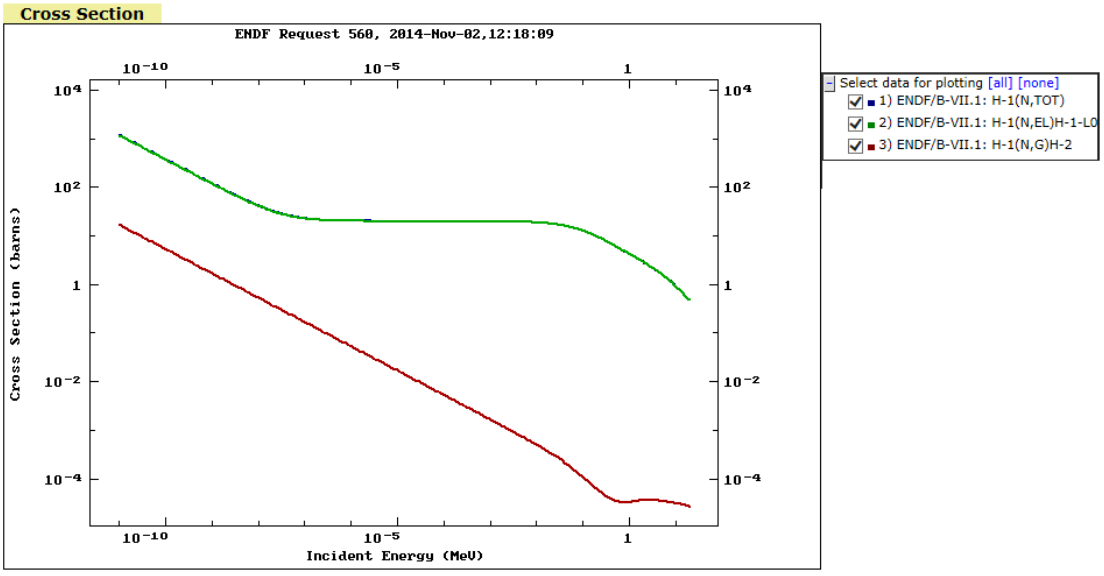


Figure 2.4. Comparison between total numbers of reactions, elastic reactions and radioactive capture reactions

Figure 2.4 is confirmed in the Evaluated Nuclear Data File (ENDF) obtained from National Nuclear Data Center (NNDC) [20]. This center includes a core nuclear reaction database containing evaluated cross sections, spectra, angular

distributions, fission product yields, thermal neutron scattering, photo-atomic and other data, with emphasis on neutron-induced reactions. Information used comes from libraries as ENDF/B-VII.1 (USA, 2011), ENDF/B-VII.0 (USA, 2006), JEFF-3.2 (Europe, 2014), JENDL-4.0 (Japan, 2010), JENDL-3.3 (Japan, 2002), CENDL-3.1 (China, 2009), ROSFOND (Russia, 2010), ENDF/B-VI.8 (USA, 2001) and ENDF/B-V.2 (USA, 1994) [20].

According to results obtained in shielding calculation inside aircraft volume, all the interactions of a neutrons from the range 1 to 20 MeV against any of the nucleus proposed as homogeneous volume of aircraft (isotopes of Fe, Al, Ca, N, O, C, H, Mg, Cl, K…) do not increase the flux of neutrons in that range of energies, 1 to 20 MeV. In other hand, the initial neutron energy spectrum is reduced due to elastic scattering interaction of neutron with different nucleus. In those reactions, those neutrons loose kinetic energy generating neutrons with lower kinetic energy.

Figure 2.5 shows the neutron energy distribution calculated inside aircraft from a source of hi-energy neutrons within energies between 1 to 20 MeV. Thermal neutron flux appeared inside aircraft increases based on two factors, neutrons scattered from the source of hi-energy neutrons simulated and neutrons scattered from keV neutron range. By means of these scattering interactions, neutron energy spectrum varies inside aircraft. They reduce the proportion of hi-energy neutrons by increasing the neutrons at lower energies. Thermal neutron (<1eV) fraction increases up to 46% while hi-energy neutrons are reduced to 9% (1-10MeV) and 26% (>10MeV) respectively inside aircraft volume, [21].

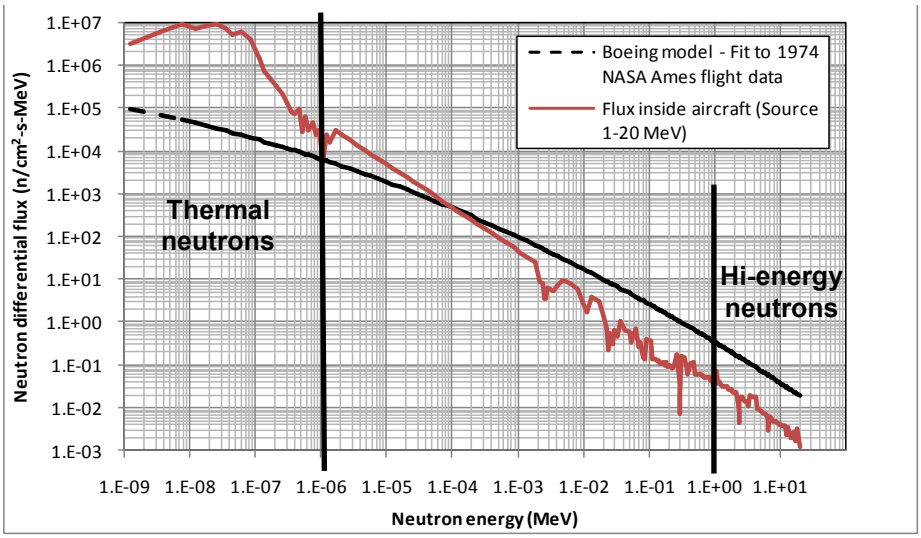


Figure 2.5: Neutron differential flux calculated inside aircraft due to scattering interaction from an external source of neutron 1-20MeV

Neutron themselves do not create direct ionization in silicon since it is neutral. It strongly interacts with nuclei of materials of electronics matter. Several ions are generated under a probability basis depending on the incident neutron energy, direction and ion shower created. Lighter particles, as α, are dangerous around their Bragg peak which could occur several tens or hundreds of microns away from the generation point. The main effects are divided in two groups depending on incident neutron energy [17]: Hi-energy neutrons and Thermal neutrons interactions.

First group of neutron interactions is based on Hi-energy neutron against Silicon nucleus of matter as represented in Figure 2.6. Silicon matter is composed by the sum of the three isotopes weighted by their relative abundance, being 92,2% of composition based on ^{28}Si (^{29}Si is 4,6% and ^{30}Si is 3,1%). The absorption of neutron by silicon nucleus, ^{28}Si, forms the excited nucleus ^{29}Si, causing probabilistically the restructuration of nucleons. Kinetic energy is shared between fragments and momentum is conserved, thus fragments are emitted in opposite directions. MonteCarlo simulations [16] reach probabilistic reactions of fragments obtained from n-Si interactions at energies lower than 1 MeV, as shown in figure 2.6.

- **n + ^{28}Si → ^{25}Mg + α**
- **n + ^{28}Si → ^{27}Al + d**
- **n + ^{28}Si → ^{24}Mg + α + n**
- n + ^{28}Si → ^{28}Al + p

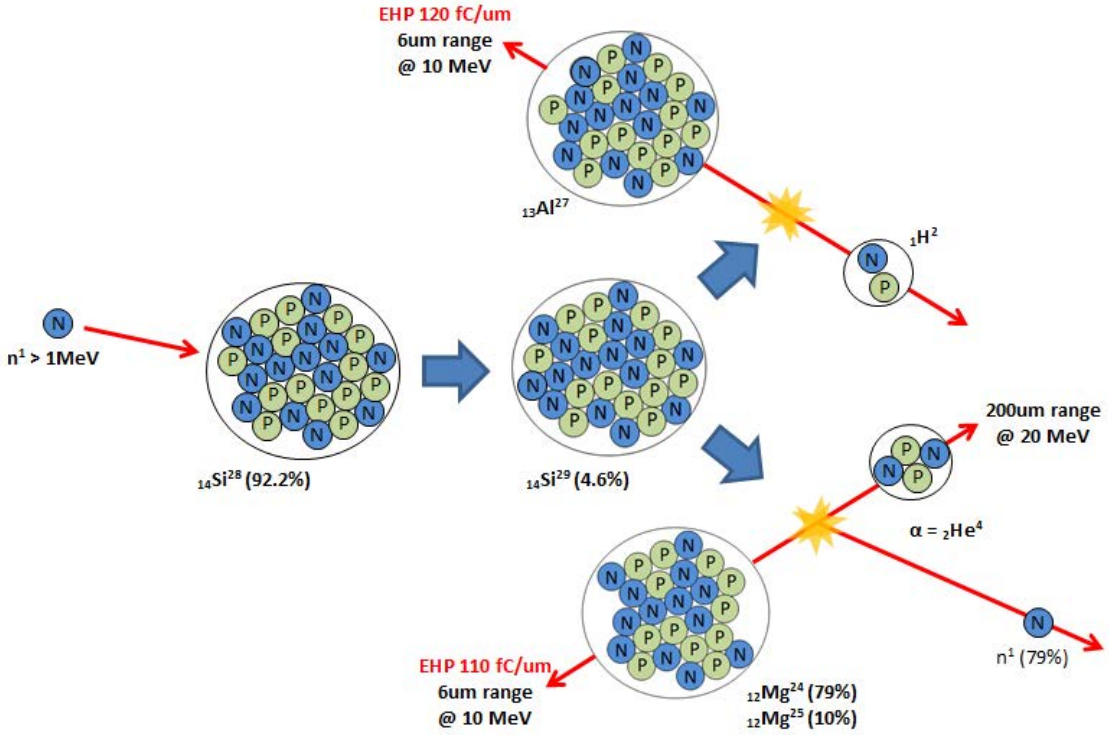


Figure 2.6: n-Si fragments

Many reactions are possible and various particles can be emitted. Particles with contributions higher than 10% of the total production are represented in Figure 2.6 with an ion shower limited to two particles. The variety and number of fragments created in absorption reaction depends hugely on incident neutron energy [10], [16], and [24]. Ion shower could involve probabilistically up to nine particles [16], [20], and [24].

Figure 2.7 represents the distance range and LET at different energies of the fragments created with incident neutrons in Si. They are calculated based on the Ziegler's range tables [8] for 100 MeV incident neutrons. The average energy of fragments created, Mg and Al, is approximately 2,6 MeV with a range of 3μm at 80 fC/μm, increased up to 6μm and >100 fC/um at 10 MeV. These higher energy fragments can be generated relatively far from the sensitive volume of cell, then travel some microns within Si^{28} matter, having enough LET to deposit energy enough to upset the cell.

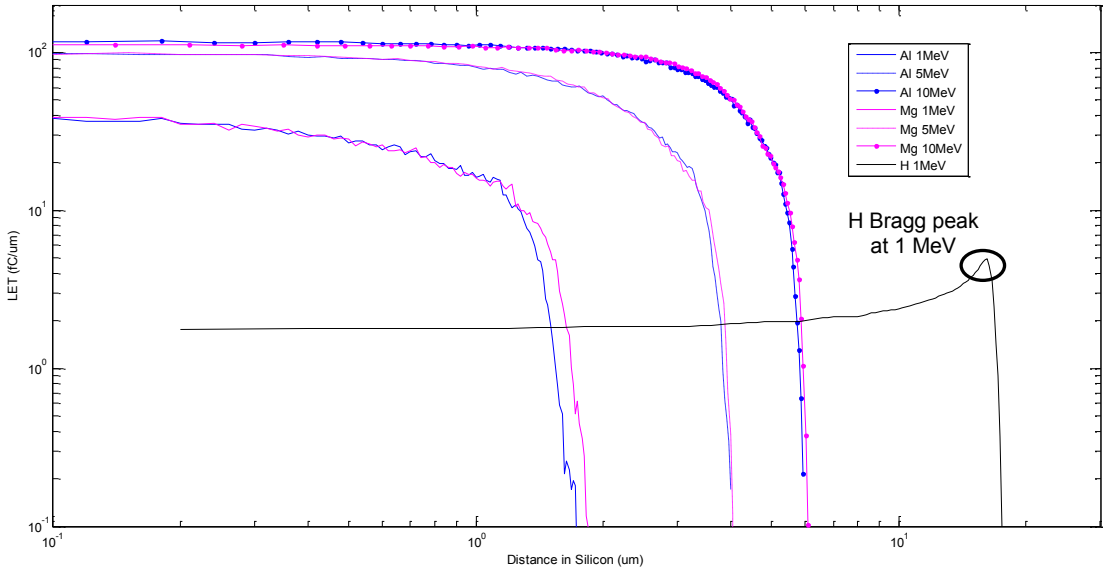


Figure 2.7: Maximum range (μm) and LET deposited (fC/μm) calculation in silicon matter from Hi-e neutrons.

Figure 2.8 is obtained through Evaluated Nuclear Data File (ENDF) obtained from National Nuclear Data Center (NNDC) [20] in same way as Figure 2.4. The cross-section curve of ^{28}Si (probabilities of iteration) with a neutron shows the low probability of radiation capture of incident neutron in full range of energies because the most probable iteration is based on elastic scattering in which neutron lose energy to ^{28}Si nuclei reducing the kinetic energy of neutron. At high energy of incident neutrons, in range of 1 to 10 MeV, it is also shown the n-Si iterations commented before in which other isotopes of different components are obtained as $^{25}Mg + \alpha$, $^{28}Al + p$, $^{27}Al + d$, $^{24}Mg + \alpha + n$.

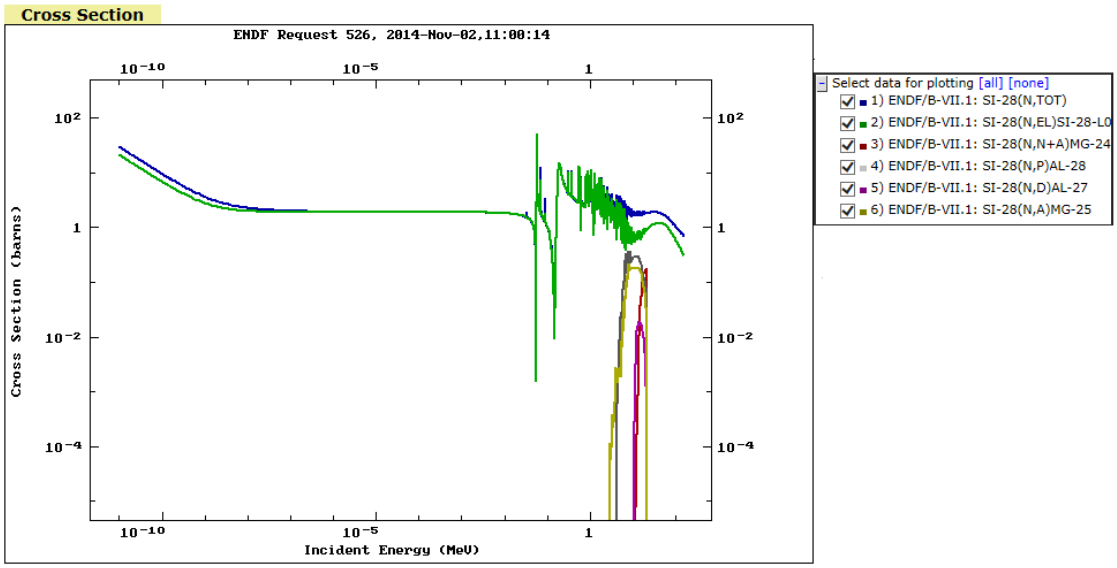


Figure 2.8: Cross-section curve of ^{28}Si

Second group of neutron interactions is based on interaction of Epithermal and Thermal neutrons (<1eV) scattered inside aircraft volume. They are not related to Silicon nucleus, but interact with impurities of silicon matter. Trace of radioactive materials contamination, [23], [25], in IC package or wafer fabrication processes as Uranium, Thorium, Boron are “fissile nucleus” being possible the radioactive capture. Most important “impurity” used during electronics manufacture processes is Boron. Boron was used extensively in electronics for borophosphosilicate glass (BPSG) dielectric layers to reduce the temperature of reflow process and as constituent of die and also as p-type dopant [17].

Due to natural Boron is found naturally in two isotopes: B^{10} (19.9%) and B^{11} (80.1%). Recent technologies have removed the BPSG and have tried to reduce the content of isotope B^{10} by a B^{11} purification process in rough materials. After BPSG removal from aerospace IC’s, current manufacturing processes are not able to remove all B^{10} isotopes and a semiconductor matter with low concentration of B^{10} is the closest process to pure B^{11} matter. This low B^{10} concentration matter reduces the B^{10} interaction rate, but involves an expensive and complex manufacturing process at the foundry. The most probable reactions obtained from n-B^{10} interaction with thermal neutrons are shown in Figure 2.9 and 2.10, [8], [9], [20], [23]:

- n-B^{10}, neutron interactions at energies lower than 1 MeV:
  - **$n + B^{10} \rightarrow Li^7 + \alpha\ (He) + \gamma$**
  - **$n + B^{10} \rightarrow Li^7 + \alpha\ (He)$**

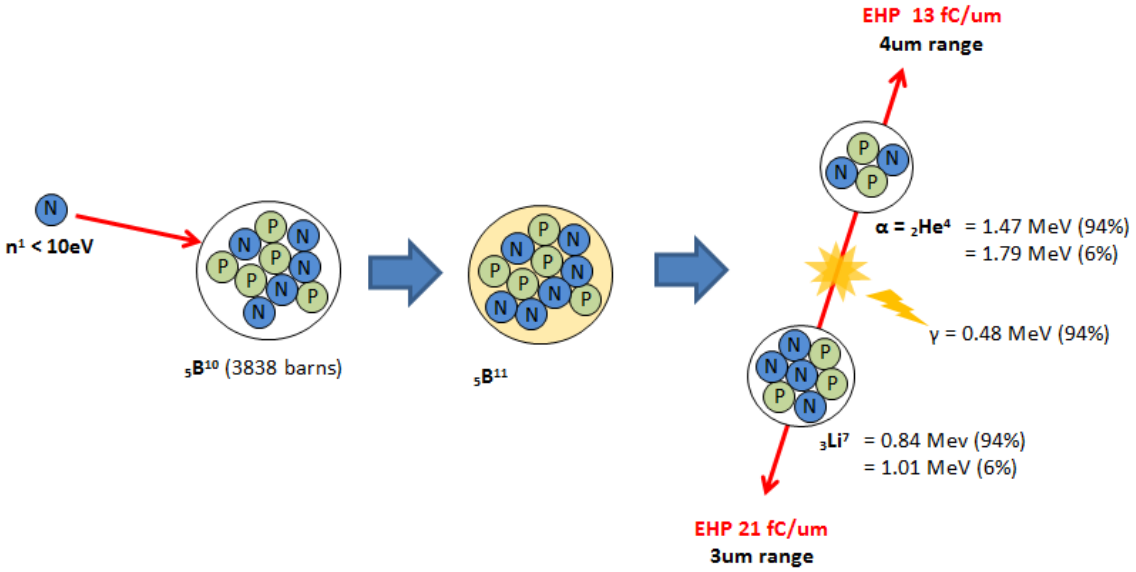


Figure 2.9: n-B^{10} fragments obtaining isotope Li^7 and He

- n-B^{11}, neutron interactions at energies close to 10 MeV:
  - $n + B^{11} \rightarrow Li^7 + \alpha + n$
  - **$n + B^{11} \rightarrow B^{10} + 2n$**
  - $n + B^{11} \rightarrow Be^{10} + n + p$

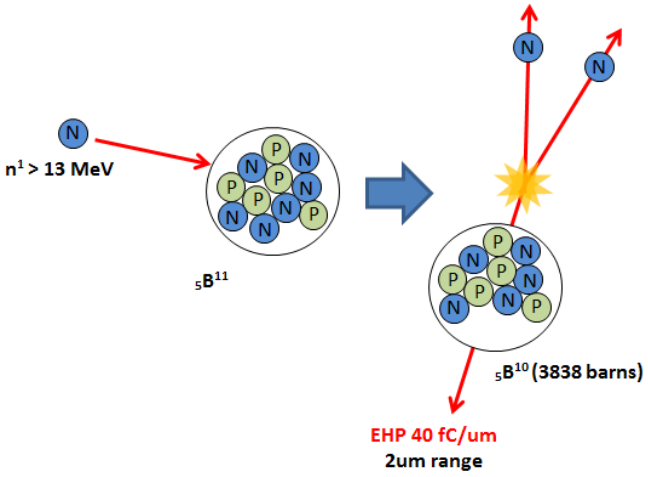


Figure 2.10: n-B^{11} fragments obtaining isotope B^{10}

According to IEC62396-5 [2], total SEU rates in devices should be calculated from the SEU rate calculated from Hi-energy neutrons multiplied by a factor to take in account the effect of thermal neutrons. In case of devices that contain BPSG, this factor could vary within a range 1 to 10. Other estimations [17] from

experimental results approximates this factor lower than 2, for devices with no content of BPSG but using rough materials containing natural B^{10}. Pure B^{11} matter (very low concentration of B^{10} isotope) reduces more this factor.

Thermal neutron absorption cross section of B^{10} is extremely high in comparison to most of other isotopes presented in semiconductor matter. Figure 2.11 also shows the Ziegler's range tables [8] for α (He) and Li in Silicon obtained from B^{10}-n reactions. Alpha particle and Lithium are emitted in opposite directions to conserve momentum. Li^7 is emitted at 1MeV with a LET of 21 fC/μm with a range of 3μm while alpha particle is emitted with a LET of 13 fC/μm with a range of 4μm. To define if a cell is upset, it is necessary that secondary fragment tracks cross the sensitive volume of cell and deposits enough energy to upset the cell, SEU

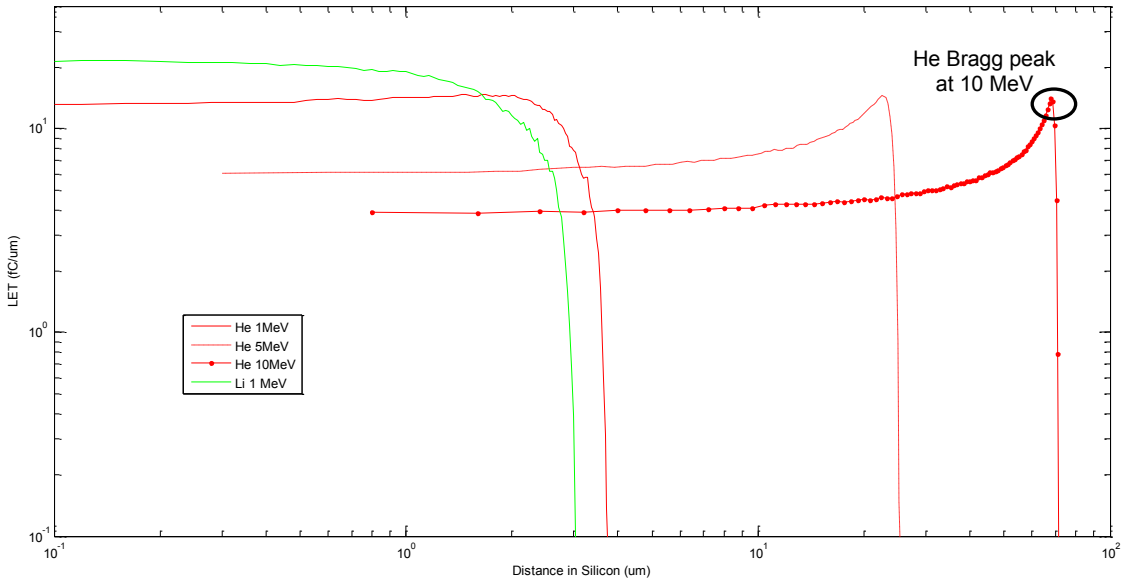


Figure 2.11: Maximum range (μm) and LET deposited (fC/μm) calculation in silicon matter from Thermal neutrons.

Figures 2.12 and 2.13 represents the cross-section curve of B^{10} (probabilities of iteration) with a neutron. They are also obtained through Evaluated Nuclear Data File (ENDF) obtained from National Nuclear Data Center (NNDC), [20]. They show the high probability of neutron capture dominates in range of energies below 1 keV, shown the n-B^{10} iterations commented before in which other components are obtained as Li^7 + α. In range of energies above 1MeV the most probable iteration is based on elastic scattering in which neutron yields energy to B^{10} nuclei reducing the kinetic energy of neutron. Other reactions could be also obtained with lower probabilities as Be^{10} + p and Be^9 + d (H^1).

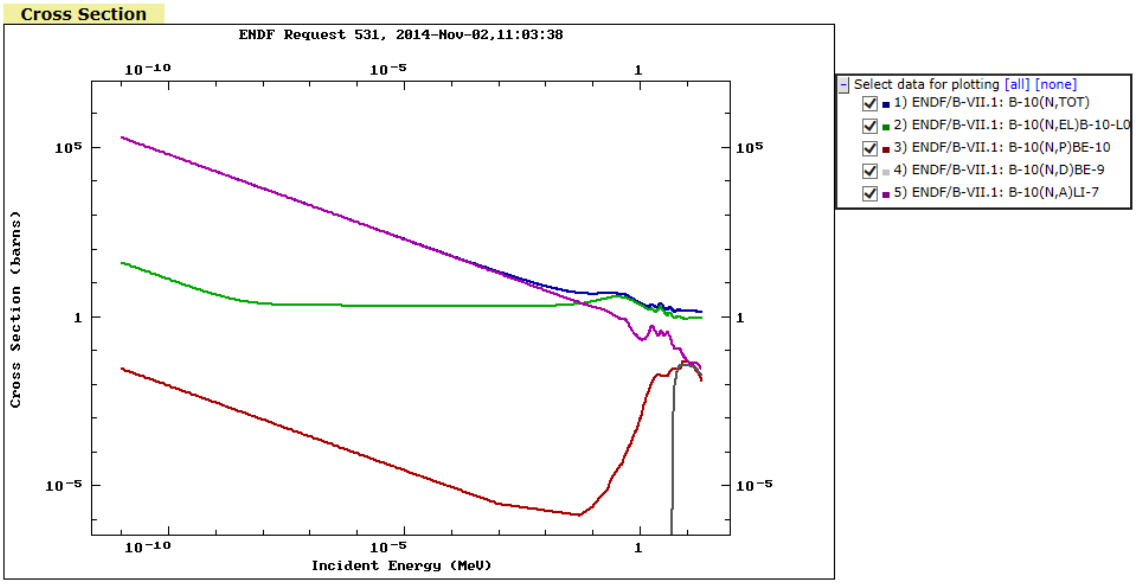


Figure 2.12: Cross-section curve of B^{10}

The cross-section curve of B^{11} (probabilities of iteration) with a neutron is quite different as the cross-section curve of B^{10}. Neutrons with incident low energies cause the nuclear capture in B^{10} while B^{11} experiments a scattering interaction where neutron loose energy in the collision. B^{11} shows a very low probability of neutron capture in the range of energies below 10 MeV where the most probable iteration is based on elastic scattering in which neutron loose energy to B^{11} nuclei, reducing the kinetic energy of neutron. Furthermore it is also shown that the most probable n-B^{11} iterations with neutrons with high incident energy, above 10MeV, create other components as B^{10} + 2n, Li^{7} + α and Be^{10} + p.

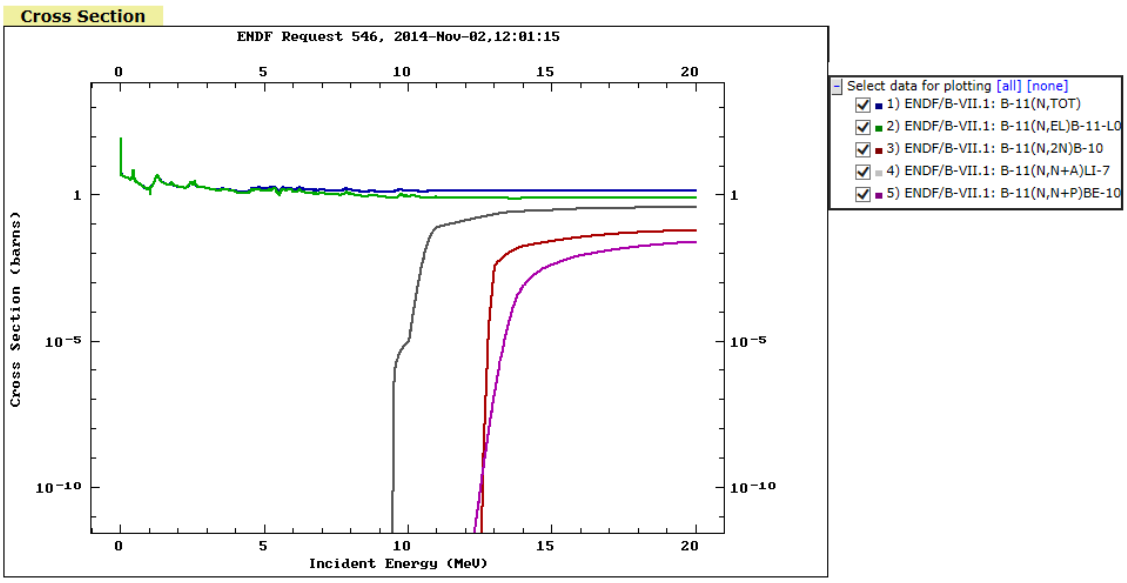


Figure 2.13: Cross-section curve of B^{11}

2.1.1 SINGLE EVENT UPSET (SEU)

The first failures due to the effects of cosmic rays were observed in electronics in 1975. This kind of radiation has been monitored at A/C flight altitudes since 1992, [25]. Their effects are related to SEU (Single Event Upset) that is part of Single Event Effect (SEE) that is a pure Hardware issue as it occurs when a bit is upset due to ionizing particle impact.

After nuclear absorption of high energy particles in Silicon medium in airborne electronics, charged fragments are emitted simultaneously and deposit LET in matter. As commented in previous paragraph, a cell is upset if nuclear fragment tracks cross cell sensitive volume and deposits enough energy to upset the cell. Downscaling evolution has two contradictory effects: less charge to upset but smaller sensitive volume. Fragments could achieve longer distances keeping enough charge to upset but the probability to impact is lower because sensitive volume is also reduced, then SEU rates evolution is not easy to predict. A clear effect of this downscaling evolution is that intercell distances between sensitive volumes (μm order) will be lower with technology integration [9], [15]. This implies that fragments from n-^{28}Si or n-B^{10} reactions, Figure 2.6, 2.9 and 2.10, could upset multiple cells, increasing strongly MCU probability. In n-^{28}Si reactions, [10], [11], Mg and Al isotopes are clearly the main contributors to MCU while for n-B^{10} reactions [12], the natural boron doping level determines the effect.

SEU could occur in the memory devices that contain the executable program or Configuration Code of FPGA causing a bug in functionalities. In SRAM-based FPGA, memory is divided into two blocks which are the Configuration-code and the User-programmable code. Configuration-code is used to specify the LUTs (Lookup tables), routing connection blocks and I/Os into FPGA while User-programmable code is the logic that keeps circuit functionality. The main share of memory, more than 90%, is routing connections. Since all resources of these devices are all susceptible to SEU, if an unintended instruction is commanded in avionics, it could cause a potential catastrophic or hazardous condition. Therefore a redundant technique must be adopted to all of them. The main problem is related to the possibility that the system apparently could continue working even though some bits have changed its state (hidden failure), which means a different configuration performing unintended activities. Consequences of these effects, caused in radiated Hardware, are listed as follows:

- Spurious signal or voltage, induced by the deposition of charge by a single particle that can propagate through the circuit path during one

clock cycle. It is a non-destructive effect as transient errors that cause a temporary upset called Single Event Transient (SET).

- “Soft” permanent hardware error that causes a change of memory cell value, called SEU, Single Event Upset. It is “Soft error” because it can be corrected including appropriate mitigation means. Single event upset (SEU) is the most common type of a single event effect. SEU is caused by the deposition of charge in a device by a single particle that is sufficient to change the logic state of a single bit from one binary state to the other.

- “Soft” permanent hardware error that causes several changes of memory cells, called multiple bit upset (MBU) refers to multiple bits being upset during the same SEE interaction by one fragment or by several fragments simultaneously. MBU differs from multiple cell upset (MCU) in which two or more bits (cells) are upset, usually bits physically located near each other, but not necessarily in the same logical word.

- Permanent destructive effect due to the rupture of the bit called Single Event Burnouts (SEB), “hard error”. Burn out of a powered electronic component as a result of the energy absorption triggered by an individual radiation event.

- Single event functional interrupt (SEFI) refers to a SEU in a device, usually a complex device, for example, a microprocessor, such that a control path is corrupted, leading the part to cease to function properly, for example the bit that controls important downstream operations.

- Permanent destructive effect called Single Event Latch-up (SEL) depending on exposed time in which transistors are kept latched in electrical short-circuit causing thermal damage. The latched path will persist until power is removed from the device, so power shall be recycled to restore normal operation. In case mitigation technique against SEL is not intended, SEL should be considered a “hard error”.

Traditionally these effects are related to volatile RAM memories because the non-volatile Flash ROM memory has been very tolerant to the neutrons in atmospheric radiation because the bit construction does not depend on the critical charge of the node. However at feature sizes, Flash memories start to become susceptible to various SEE, including SEFI. Total ionizing dose (TID) effects refer to the cumulative effect of ionization in to bit physical architecture. Flash memory device leading to a gradual degradation of electrical parameters.

Ionizing dose is contributed by all of the major particles that constitute the radiation environment within the atmosphere as neutrons.

Several experiments have verified the fairly linear relationships between flux, critical charge of cell and operational frequency versus error rate. These are the three factors that state a fundamental limitation in bit scaling Moore's law (reduction of bit size and storage charge to increase bit density). The most important way to analyze how SEU can affect design is given by the fairly linear relationship between the flux and error rate, [2]:

$$\boldsymbol{SEU\ rate = Flux \cdot \sigma \cdot Nbits} \qquad \text{Eq. 2.1}$$

Where Flux is defined as the atmospheric differential flux at which particles hit an electronic area over Energy spectrum (MeV), measured as particles/cm^2/s.

“Plateau cross section” determined empirically, represented as σ, is defined as the SEE susceptibility of an electronic device to ionizing radiation. Cross-section σ in radiation terms for proton and neutron interactions is the combination of sensitive area and probability of an interaction depositing the critical charge for a SEE. The cross-section may be calculated using the following formula: σ = number of errors / particle flux. This data should be obtained by an experimental test in each component along a wide range of energy transferred to the device, because it highly depends on the internal design of bits. The units for cross-section are cm^2 per device or per bit.

Approximate values for current technology, extracted as an average of Weibull distributions of different components, are given for a rough analysis as:

$$\sigma\text{HEAVY ION} \approx \text{between } 10^{-7} \text{ and } 10^{-6} \text{ cm}^2\text{/bit}$$
$$\sigma\text{NEUTRON} \approx 10^{-13} \text{ cm}^2\text{/bit}.$$

The total amount of bits saved in memories on board in an aircraft is the key characteristic to determine the problem caused by neutron flux and its SEU rate induction.

2.1.2 MULTIPLE BIT UPSET (MCU)

After nuclear absorption, charged fragments emitted simultaneously deposit LET in matter. MCU appears when each fragment upsets a single cell simultaneously, as Type 5 shown in Figure 2.14, or if only one fragment upsets several cells, as Type 2, 3 and 4. MCU could be usually found in bits physically located near each other, but not necessarily in same data word.

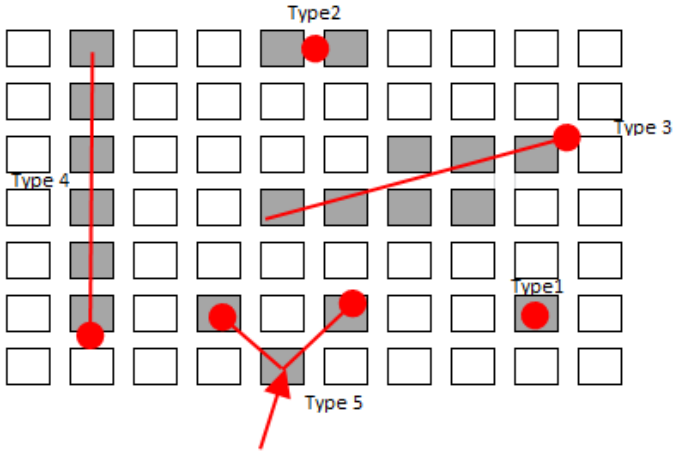


Figure 2.14: Typical MCU patterns

A clear effect of this downscaling evolution is that intercell distances between sensitive volumes (µm order) will be lower with technology integration [9], [15]. This will imply a clear increment of ratio MCU/SEU will be then expected due to several charged fragments are emitted simultaneously being each fragment able to upset a single cell simultaneously or if only one fragment upsets several cells due to intercell distances are highly reduced.

As commented in previous paragraph, a cell is upset if nuclear fragment tracks cross cell sensitive volume and deposits enough energy to upset the cell. Device downscaling evolution has become a concern because reduces the critical charge required to upset the cell due to voltage supply and cell volume capacitance reduction.

Critical charge is the smallest charge that keeps the bit architecture stable in one of the two possible states. If a charge higher than the critical one for certain bit architecture is injected or deposited by SEE in the sensitive volume of cell, there is a high probability that the bit architecture become unstable changing the value stored. For many devices, the unit applied was the picocoulomb (pC). Due to scaling evolution, the critical charge in new small geometries in bit design architecture is measured in femtocoulomb (fC). Simulations, [9], [15],

[16], [18], of intercell layouts for modern technologies fabricated in 130 and 250nm feature size estimate a reduction of critical charge to upset the cell to 11-17 fC. In 65 to 90 nm feature size technology, it becomes smaller to 1-7 fC.

According to [13], [14], [35], [36], and [37] MCU patterns (Row x Column) are characterized by the number of bits effected in spatial distribution in rows and columns in memory. Most events are identified as 1x1, SEU shown in Type 1 in Figure 2.14, but the total number of events in MCU are relative large. It affects to many rows and few columns of a memory structure depending on neutron incidence angle. Different patterns are found as 1x2, 2x1, 2x2, 2x4, 3x2 and 4x1, [12], marked in Figure 2.14 as Type 2, 3 or 4. MCU exhibits a strong dependence on device orientation as represented in Figure 2.15, because increases the total number of bits in patterns at 79º neutron incidence angle [22], as Type 5 in Figure 2.14. Therefore, maximum range estimated in a conservative way (maximum number of bits affected in line) is 5 bits while maximum number of bits affected estimated (sum of all bits affected between rows and columns) in MCU is up to 12 bits, [10], [12], [13]. As per IEC62396-1 [2], for future devices with feature sizes <35nm, a 100% of SEU/MCU fraction is expected being highly affected by angle of incidence. MCU fraction could increase from a 30% at 45º up to 80% at 90º incidence. An example of SEU/MCU fraction depending on elevation incidence angle is shown in Figure 2.15.

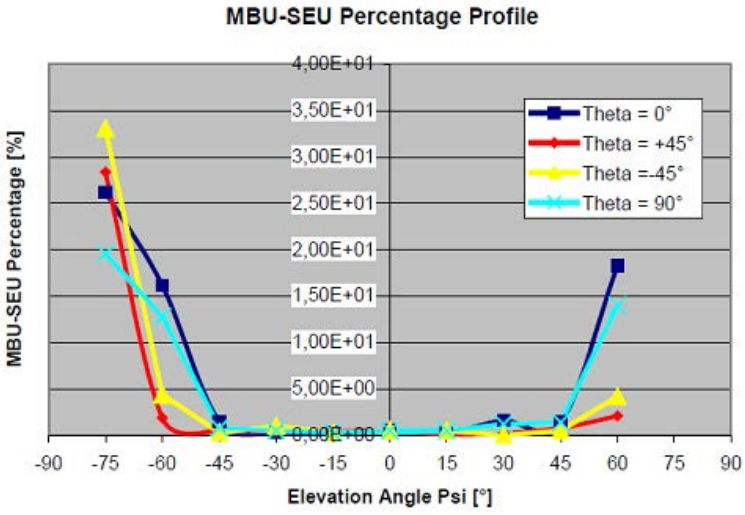


Figure 2.15. MCU/SEU fraction in a SRAM device as a function of neutron angle of incidence

2.2 CURRENT SOLUTIONS IN AIRBORNE ELECTRONICS

As described in previous paragraphs, SEEs could be a “soft” permanent error that causes a change of memory cell value. This error could be exhibited in its single effect, SEU, or in its multiple effect, MCU. State-of-the-art of solutions to mitigate the effects of SEU/MCU is divided into categories: reconfiguration-based and redundancy-based solutions. First ones restore as soon as possible the original values in memory after SEE event while second ones mask SEE effects to the circuit outputs via redundancy with the purpose to remove all single points of failure in the circuit.

Redundancy-based techniques are not intended to remove SEU events. These methods mask SEE effects to the circuit outputs via redundancy with the purpose to remove all single points of failure in the circuit mitigating in this way the propagation to the circuit outputs. In some non-safety critical applications, it is sufficient to detect an error caused by SEU and just flag whether there is data affected, invalid or corrupted. One method is the duplication of functionalities, [7], [28], [31], where the system could detect errors based in outputs monitoring comparison. This method is able to detect that a function is in fault but there is no possibility to detect which one is in fault, because there is no evidence about which logic is working properly.

When SEU fault masking is mandatory in safety critical applications, a fault tolerant system should be implemented. That means that only detection capability is not enough, and correction capability is required. In fault tolerant systems, most common architecture is TMR, triplication of functionalities, where monitor comparison [28], [31], typically voting circuit, could also discriminate which function is in fault. These techniques come with high area overheads due to TMR design and also require a coupled reconfiguration-based technique to support the soft error accumulation removal in memory.

Reconfiguration-based solutions try to restore as soon as possible the original values in memory after SEE event. These methods introduce a resources overheads corresponding to the circuit required to control the periodically bitstream rewritten procedure.

State-of-the-art of different FPGA suppliers allows the partial reconfiguration to rewrite only the selected part without stop the circuit implemented in device for a shorter period of time. Readback is a post-configuration read operation of the configuration memory used to verify that the status design downloaded is right. Readback is the process of reading all the data in the internal

configuration memory while Partial Reconfiguration is a post-configuration write operation to the configuration memory.

FPGA technology provides the flexibility of on-site programming with Partial Reconfiguration. It is based on the modification of an operating FPGA design by loading a partial configuration file, usually a partial BIT file, as shown in Figure 2.16. Partial BIT files can be downloaded to modify reconfigurable regions in the FPGA without compromising the integrity of the application

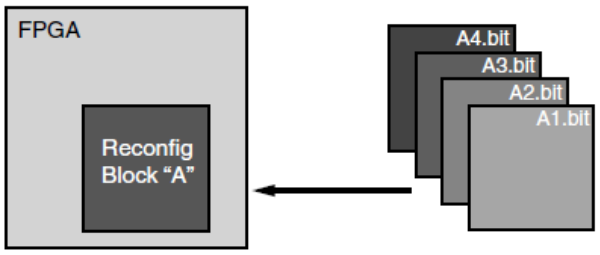


Figure 2.16. Partial reconfiguration BIT files

A simpler method to SEU correction is to omit readback and detection of SEUs and simply reload the entire Frame segment at a chosen interval. This is called "scrubbing." Scrubbing requires substantially less overhead in the logic system, but does mean that the configuration logic is likely to be in "write mode" for a greater percentage of time. Furthermore, it is required to implement a hard-copy of the data to use in the reloading process.

The more traditional method of verification of the data stored in configuration memory is to readback the data and performs a bit for bit comparison. This requires the use of the original file and the readback file which are equal in size to the original bit-stream used to configure the FPGA. This method would effectively require triple the amount of system memory to detect SEU events.

Most popular technique is EDAC based on Single/Multiple-error correcting techniques, as Hamming, BCH, R-S, Berger codes that reduces the resources required. The most important drawback of multi-error correcting codes is that their logic complexity increase exponentially as range of maximum error correction capability, Figure 2.17. This topic is an important factor to take into account when architecture is designed for MCU correction.

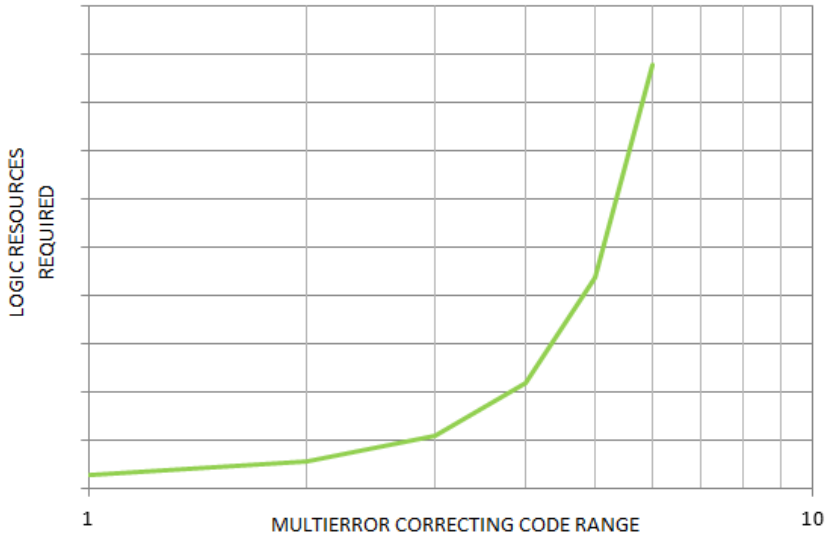


Figure 2.17. BCH resources required as error correcting range increases

Some reconfiguration-based and redundancy-based strategies are listed below:

1. Parity bits (PAR), it is a low latency process able to detect an odd number of bit errors. A parity bit could be added in the original binary code indicating if the total number of bits with the value one is even or odd. Parity bits are used as the simplest form of error detecting code. This technique fails to detect a failure when there is more than one error in data [34].

2. Double Modular redundancy (DMR) is also a fault-tolerant redundancy based on dual modular copies of system. Double Modular Redundancy (DMR) is also used based on duplication of data. Error detection is based on a voting comparison between the outputs generated by the different copies of same code [26], [28]. This kind of system is able to detect errors in any of the copies but not to detect which one is in fault. This technique also requires a coupled reconfiguration-based technique to support the correction of soft error and the error accumulation in memory.

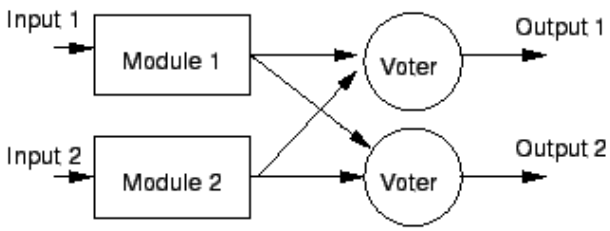


Figure 2.18. DMR scheme

3. Triple Modular redundancy (TMR) is a fault-tolerant redundancy based on modular copies of system. Three systems perform the same process and

those results are processed by a majority-voting system to produce a single output. If any one of the three systems fails, the other two systems can detect which system is fault and mask the fault to system outputs. This technique also requires a coupled reconfiguration-based technique to support the correction of soft error and the error accumulation in memory.

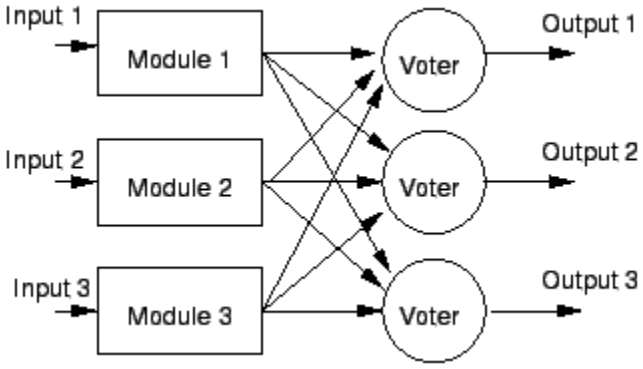


Figure 2.19. TMR scheme

Error detection is based on a voting comparison, low latency process, between the outputs generated by the different copies of same code [7], [28]. Some communication systems use N-modular redundancy as a simple form of forward error correction. For example, 5-modular redundancy communication systems use the majority of 5 samples, if any 2 of the 5 results are erroneous, the other 3 results can correct and mask the fault.

4. Single Error detection and Correction codes (SEC) as Hamming codes, [5], [6], [7], [28]. An error-correcting code (ECC) is a system that adds redundant bits to the original bitstream, being capable to recover even when a number of errors (up to the capability of the code being used). Error-correcting codes are usually distinguished between convolutional codes and block codes:

   - Linear Block codes. It is an error-correcting code for which any linear combination of codewords is also a codeword. Most typical application of these linear block codes are the parity. The principle to encode is based on a generator identity matrix with by the combination of two or more codewords.

     Hamming codes pertains to the linear error-correcting codes that can correct one-bit errors in the complete bitstream. An improvement of Hamming code is the Extended Hamming code that has the capability to correct one error but also to detect two errors by adding a parity bit to the Hamming code

- Convolutional codes. It is an error-correcting code that process on a bit-by-bit basis. Convolutional codes unlike systematic block codes, the sender does not send the message bits followed by the parity bits. In a convolutional code, the sender sends only the parity bits. The encoder uses a sliding window to calculate the parity bits by combining various subsets of bits in the window

- Cyclic codes. It is an error-correcting code based on linear codes in which all properties of linearity and the associated techniques apply equally to cyclic codes. The cyclic code includes a property in which any codeword is generated by other codeword shifted cyclically. Therefore, all codewords can be generated from a single bitstream by shifting the codeword. From a starting generator sequence, it is shifted it cyclically left until all positions are registered.

- A cyclic redundancy check (CRC) is a single-error detecting cyclic code. The generator polynomial is used as the divisor of the polynomial. Therefore, the input data is used as the dividend, and where the remainder becomes the result. CRCs are easy to implement in hardware.

5. EDAC, Error Detection and correction codes, designed to correct multiple errors (MEC) in data based on Reed–Solomon, BCH, and Berger. [26], [28], [29], [30], [31]. Code complexity increases exponentially as range of error correction capability. As higher number of error correction capability, higher consumption of resources in FPGA is requested, as adders, multiplexors and FFs. Single error correction codes as Hamming code are particular cases of these multi error correction codes where correction capability is set to one error.

    - BCH codes are cyclic error-correcting codes that are based on finite fields. BCH code is designed to correct a certain number of symbol errors. It is possible to design binary BCH codes that can correct multiple bit errors. Main advantage of BCH codes is the ease decoding, known as syndrome decoding.

      BCH codes are used in applications such as satellite communications,[2] compact disc players, DVDs, disk drives, solid-state drives[3] and two-dimensional bar codes

    - Reed–Solomon (RS) codes is a non-binary cyclic error-correcting code that could detect and correct multiple random symbol errors. An

RS code can detect any combination erroneous symbols and correct error up to half of the symbols. In Reed–Solomon coding, source symbols are viewed as coefficients of a polynomial p(x). Reed-Salomon code (RS) has been studied being recognized to be a special case of multilevel BCH codes, in which decoding methods are similar to those used for binary BCH [6].

Reed–Solomon codes have since found important applications from deep-space communication to consumer electronics. They are prominently used in consumer electronics such as CDs, DVDs, Blu-ray Discs, in data transmission technologies such as DSL and WiMAX, in broadcast systems such as DVB and ATSC, and in computer applications.

6. Watchdog timer to reset system or Boot Reconfiguration [7], [32], [33], able to detect timing and scheduling errors, as [27] based on external FPGA. A watchdog timer (WDT) is an electronic timer that is used to detect and recover malfunctions. During normal operation, the FPGA synchronously flip the watchdog timer to prevent it from delays. If, there is an error in which FPGA is not able to restart the watchdog, the timer generates a timeout signal. The corrective actions typically include a safe state or a boot reconfiguration.

7. Shielding protection increases greatly the weight of system not achieving enough attenuation. The shielding effect describes the interactions between an electron and the nucleus in any atom. Shielding effect acts as a reduction in the effective nuclear charge on the electron shower reducing the kinetic energy.

Figure 2.20 represents a comparison of State-of-the-art techniques that have been done in terms of resource consumption and scrubbing timing. Most popular techniques are DMR/TMR and EDAC for multiple error correction (MEC). TMR is an approach based on the triplication of critical functionalities for SEU/MCU masking protection. TMR triplicates each register into three different flip-flops, for that reason TMR has been placed in the top part of graph where three units of same system is required to decide the correct state of the register. TMR requires extra memory space (x3) and/or voting system, incurring extra cost, components, weight, space, dissipation power and higher dimensions for PCBs.

DMR is also an approach that is based in a copy of system as TMR. In this case it is based on the duplication of critical functionalities for SEU/MCU masking protection. DMR duplicates each register into two different flip-flops, for

that reason DMR has been placed also in the top part of graph where two units of same system is required to decide that a failure has been produced In any of both registers because it can determine that a failure has occurred but it cannot decide which one is in failure.

Parity bit is the quickest techniques to detect an odd number of bit errors. It is also one of the simplest forms of error detecting code. Main drawback of this technique is that it fails to detect a failure when there is an even number of errors in data.

EDAC architecture is an approach that reduces the resources required when compared to DMR/TMR. EDAC could be based on Single-error correcting techniques, as Hamming code, although fail to detect MCU. To solve this issue, other EDAC architectures based on Multiple-error correcting techniques (MEC), as BCH, R-S, Berger.., are used but require much more resources than SEC due to coding complexity increases as range of errors to be corrected are required.

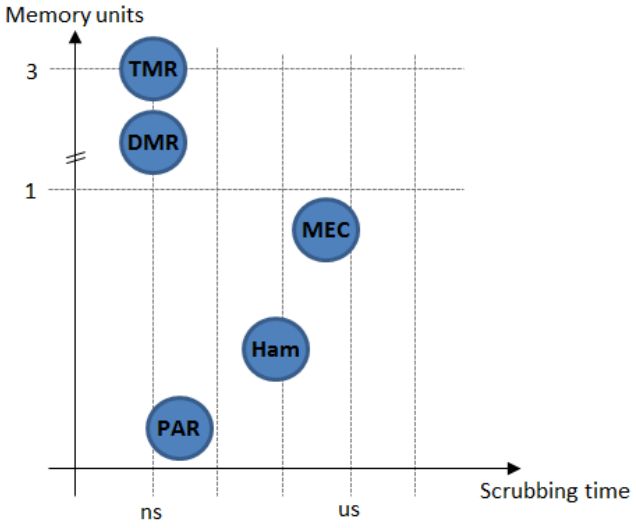


Figure 2.20. State-of-the-art of Mitigation techniques

CHAPTER 3: PROPOSED SOLUTION: ARCHITECTURE

PHYSICAL ARCHITECTURE

OPERATION ALGORYTHM

IMPLEMENTATION ISSUES

As explained in Chapter 2, the classical fault tolerant solution against SEE effects is TMR plus scrubbing. The proposed work in this Thesis equally performs under the fault tolerant approach, improving the resources involved, mainly reducing the operation time and the memory overheads, compared with the classical solution.

The proposed solution is composed of:

- The Physical Architecture.
- The Operation Algorithm.

.

3.1 BASIC PHYSICAL ARCHITECTURE

Figure 3.1 summarizes the basic Physical Architecture, composed of:

- The SRAM memory. A simplified memory of 17x8 bits is chosen as example in this text to show all the concepts. This memory is split into:
    - Most of this memory corresponds to passive data (grey bits, from A1 to H16 in our example).
    - There are also some bits which are used for the scrubbing function itself (SEC1), which is identically duplicated (SEC2).
    - There are also some bits in it which are used as the reference (from A17 to H17) for the scrubbing function (checksum and parity bits).

- Hardwired Seed Bits (HSB): this is a set of just a very few bits which are not susceptible to radiation. They have the essential information to retrieve the scrubbing reference (from A17 to H17) when required, even under radiation event.

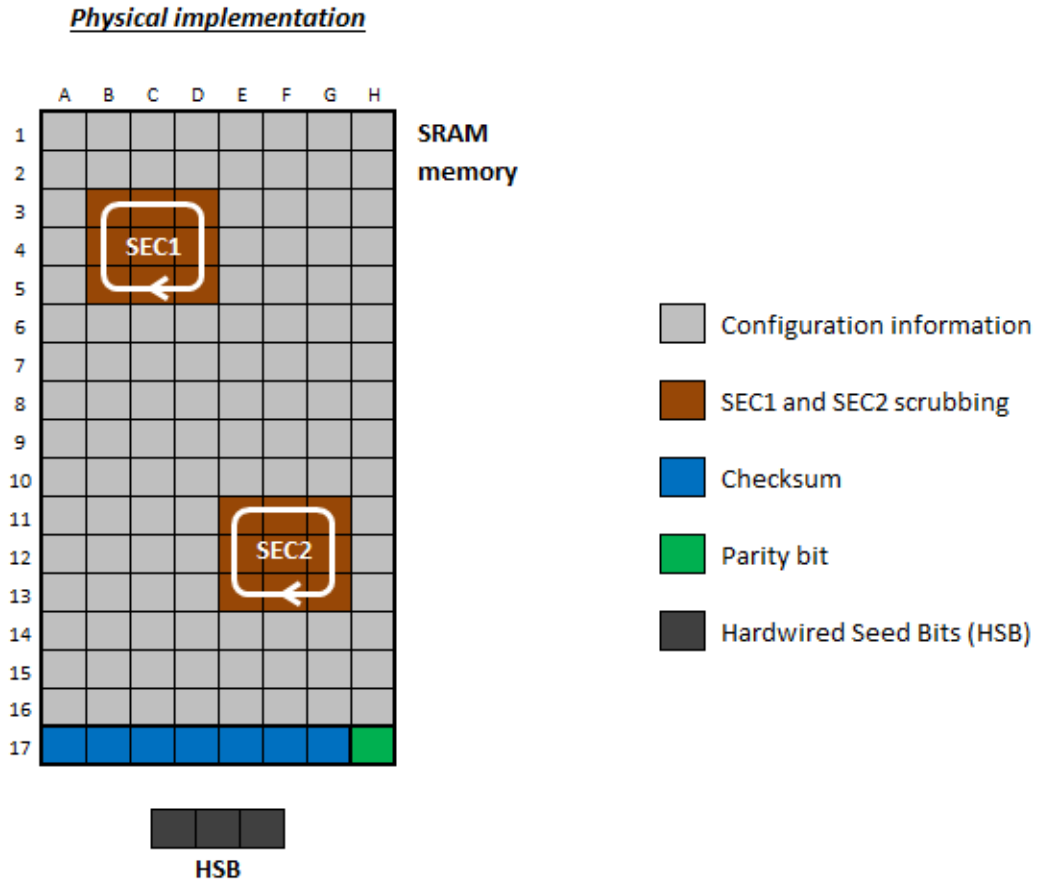


Figure 3.1: Proposed archtiecture

SEE could occur in any bit at SRAM (A1 to H17). Failures in these memories could cause a potential catastrophic or hazardous condition in aircrafts. As explained in Chapter 2, the SRAM memories (also Flash memories for Flash-based FPGAs) of Complex Electronic devices are vulnerable to the potential effects of the airborne radiation environment. This architecture would guarantee the integrity of this *passive Data* (Configuration Code of an FPGA) and, therefore, could certify these systems as safety critical applications in aircrafts, fulfilling Standards (Section 1.1.).

Furthermore, the Hardwired Seed Bits, *HSB*, are also added in the Physical implementation side as the permanent reference of architecture. This reference contains the essential information, neutral to radiation, from which errors in original data can be retrieved. There is still a risk of having an upset in the HSB, although this risk has been eliminated due to the their physical implementation. This reference is not saved physically in SRAM memory. In order to be neutral to radiation, It shall be somehow physically implemented to prevent the vulnerability to radiation, for example, pull-up resistors, ROM or a Hardened memory.

Figure 3.1 also allows a **Multilevel Architecture** operation approach. This means that checksum and parity are checked first from HSB, and then Data is checked from checksum and parity. This multilevel approach allows an significant (exponential mode) reduction ratio between the amount of data and the size of the required essential reference (HSB) for retrieving it from errors.

3.2 BASIC OPERATION ALGORITHM

The SRAM is part of a digital system in which there is also a processor (FPGA, microprocessor, DSP, etc) that executes all the functions (main functions and scrubbing algorithm). The entire memory is sequentially scrubbed, periodically inserting the scrubbing operation within the main function operation. Therefore, execution time must be shared among the functions to be performed by the processor. One of the key objectives of a proposed HSB architecture is to reduce the time consumption by decoupling the detection tasks from the correction activities.

Instead of a Triple Modular Redundancy (TMR), a Double Modular Redundancy (DMR) scrubbing function approach is applied in this Architecture. As explained after, Double redundancy is enough to meet fault tolerant requirements, thanks to the Operation Algorithm, as explained at 3.3.

Figures 3.2, 3.3 and 3.4 represent the main steps of the scrubbing procedure, where the arrows link the data to be scrubbed to its associated reference data to be used by SEC function. The operation algorithm is executed in two consecutive steps as follows:

1. **Step 1** (Fig. 3.2).
    - Target: Checking the retrieving system, guaranteeing the Checkword (checksum + parity) to be cleared.
    - Action: Comparison between Checkword and SEC syndrome computation (SEC output from HSB).

    The possible scenarios in this Step are as follows:

    1.1. An error has been identified in the Checkword level through a mismatch in the previous comparison. Scrubbing procedure is then launched to correct the found Checksum error . After this action, Checksum is healthily ready to be applied at step 2.

    1.2. There is a matching in the comparison, no error is found at the Checkword Level, which is already healthily ready to be applied at step 2.

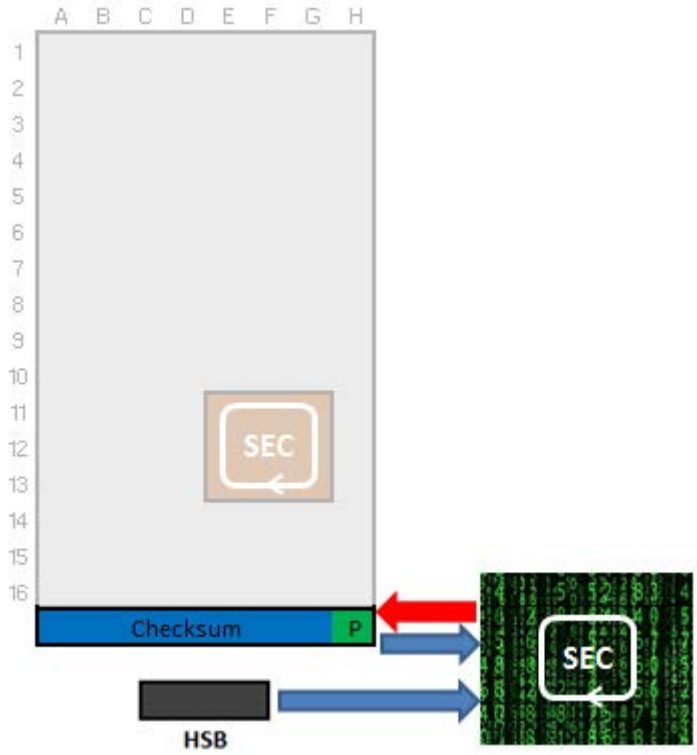


Figure. 3.2. Both SEC functions assess Checkword from HSB

2. **Step 2** (Fig. 3.3 and Fig. 3.4)
    - Target: Data Level guaranteed to be cleared
    - Action: Detection (parity function) and Correction if requested

The Checkword level is composed of the checksum bits for correction and an additional bit for a just parity error detection, which constitutes the SEC reference for the data level to be scrubbed. SEC function can be used in this case. The possible scenarios under this step are as follows:

2.1 Error Detection, Fig. 3.3: only the parity bit is used to detect possible errors at the data level.

2.2 Error Correction, Fig. 3.4, when required. If an upset is detected through parity bit, it means that there is a single corrupted bit in an unknown position within the data level. In this case, SEC scrubbing function is applied to identify and correct the error from Checksum bits.

2.3 If no error is detected, correction (step 2.2) is skipped, saving time within an optimized operation.

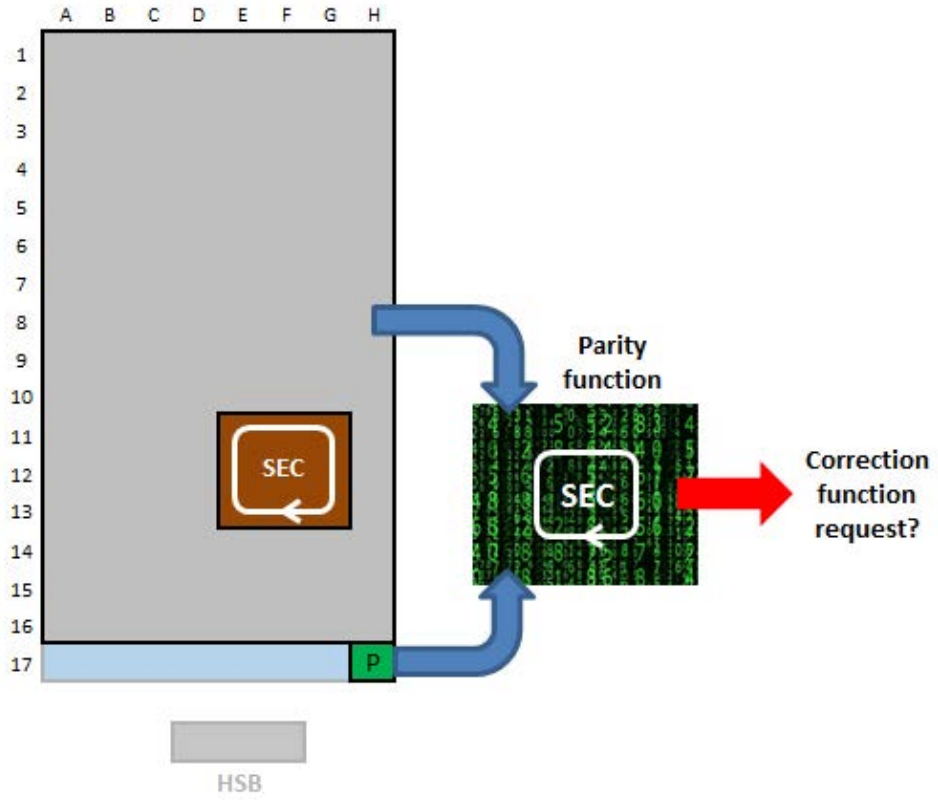


Figure. 3.3. Parity assessment of Data Level bits

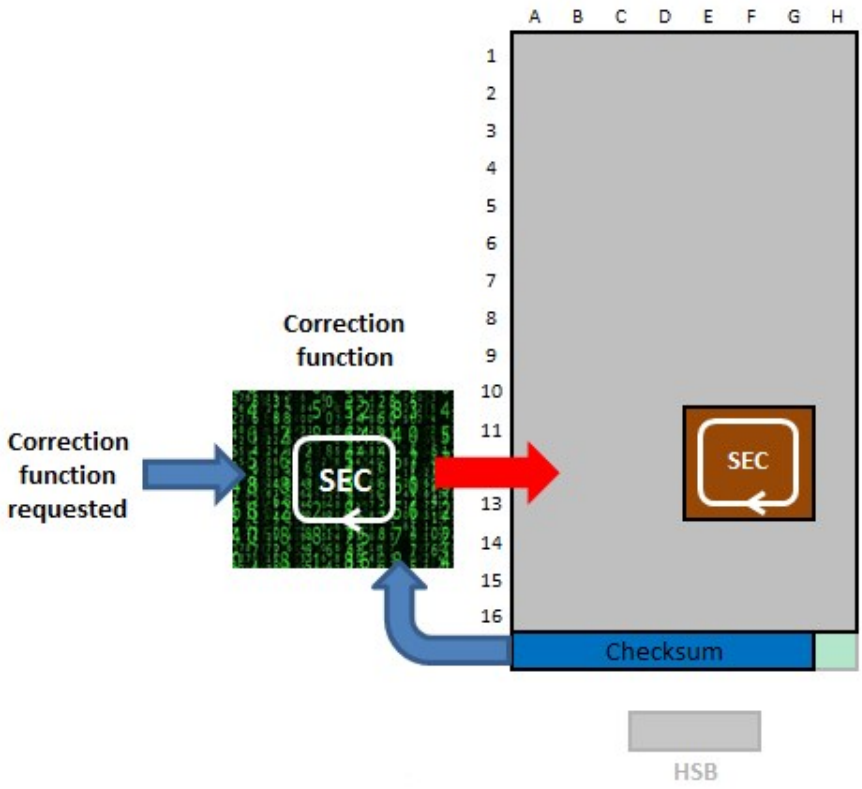


Figure. 3.4. Data level correction from Checkword data

Figures 3.5 represents the operational flux algorithm according to the steps commented before. The algorithm is run cyclically in order to keep the integrity of the memory data.

The time involved applying this process is significantly low due to the following facts:

- No errors take place in memory most of the time and just the Principal Operation line is required to be performed under no error condition. Principal Operation line is composed of just two computations, the Checksum validation and the Data Parity error detection.
- Checksum validation is also a very light time operation due to:
    - As explained later, Single Error Correction (SEC) techniques are enough in the proposed Architecture instead of Multiple Error techniques, since no more than one error is guaranteed to take place at each scrubbing.
    - SEC function is applied for just checking the Checksum bits from the HSB
- Parity error detection is a very light time operation itself. Again, Parity detection is enough since no more than one error is guaranteed for each scrubbing in this Architecture, as explained later.

Although error events are potentially dramatic from the aircrafts operation point of view, error correction takes place scarcely when compared with the no error time operation (Principal Operation Line). To illustrate it, let's consider one radiation event on memory per week as a rough ratio for a commercial aircraft. This is a risky ratio from the aircraft operation point of view since there is a crash possibility every week. However, the Error Correction time involved is negligible, less than (let's say) a second per week, while Parity and Checksum validation operation are continuously executed Therefore, the main computing time consumption worthy it to be optimized is the Principal Operation Line computation.

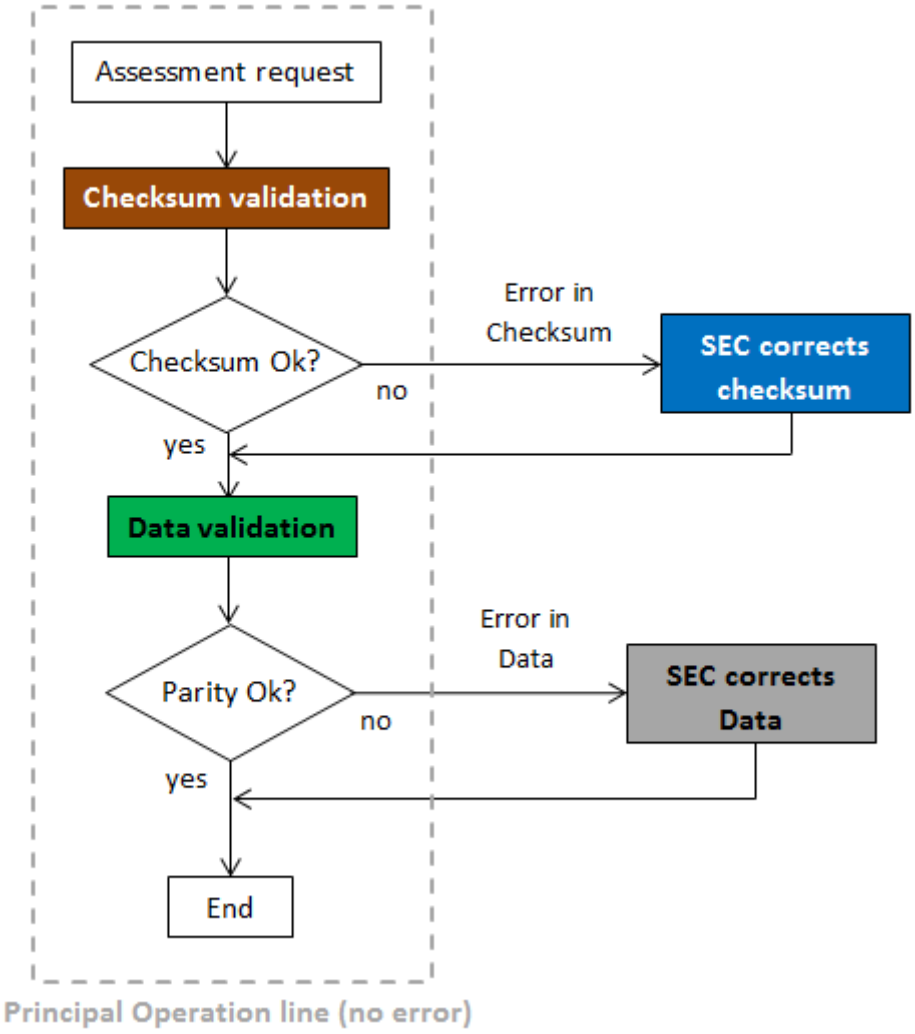


Figure. 3.5. SEU Algorithm used in architecture

3.3 ARCHITECTURE ISSUES

The basics of a system to retrieve Bit Errors in a memory have been explained in this chapter. However, different issues may be arised when aiming to a Fault Tolerant System performance, in an actual radiated aerospace environment. They are as follows:

ISSUE 1: MCU space distribution

A Single Event Effect (SEE) could lead to a MCU, which means that a set of neighbor bits have been upset due to a neutron radiation event. As explained in Chapter 2, according to [12], [13], [14], [35], [36] and [37], MCU shape patterns (Row x Column) are characterized by the number of bits affected in a spatial distribution by rows and columns in memory. The probability of having a SEU event (just 1x1) depends on the incidence angle of the radiation and, in general, the probability of having an MCU event is quite high.

Current Error Correction Techniques were explained in Chapter 2. Single-error correcting techniques are simple to use with low resources consumption (time processing and memory). However, they seem not to be suitable in this case since more than one cell might be affected at the same time. On the other hand, Multiple-error correcting techniques such as BCH could solve the problem, though they could be unpractical due to the resources involved to implement it. Eventually, TMR (Triple Modular Redundancy) is chosen as the most balanced solution so far, assuming the redundancy level cost.

The Issue that may be raised here is:

- Is this architecture able to optimize the TMR performance against a MCU?

ISSUE 2: Fault Tolerance performance

Fault-Tolerant requirements imply downsizing the error probability to below a certified threshold, normally defined as Design Assurance Level (DAL) within the Aerospace Industry. This condition means that the system must guarantee the proper operation even under a failure of any of its components.

The SECs functions stored in memories might be affected themselves by an MCU, which means that the Error Correction would not be fully guaranteed in this case. Therefore, the second Issue that may be arisen is:

- Is this architecture able to guarantee Fault Tolerant performance?

ISSUE 3: HSB Implementation

Hardwired Seed Bits (HSB) is a set of just a few bits which are not susceptible to radiation. As an example which will be explained later, 2044 data bits can be sheltered with just 4 HSB. On the other hand, these bits must be immune to radiation. Different ideas might be considered to implement them such as jumper pins (a pull-up/pull-down approach) or flash memory.

The issue to consider in this case is:

- How to physically implement the HSBs?

CHAPTER 4: PROPOSED SOLUTION: IMPLEMENTATION

VIRTUAL-FRAMED HSB CONCEPT FOR MCU CORRECTION

FAULT TOLERANT ARCHITECTURE

DETAILED CODIFICATION

This Chapter 4 gives answers to the Architecture Issues concluded in Section 3.3. (Issues 1, 2 and 3). Issue 1 consists in how to deal with Multiple Cell events, which is answered in Section 4.1. Issue 2 consists in how to deal with events in the scrubbing function itself in order to meet Fault Tolerant requirements, which is answered in Section 4.2. Issue 3 consists in how to implement the HSB, which is faced in Section 4.3.

Multiple Cell Upsets (MCU) is an issue that has to be dealt with when designing electronics for working in a radiated environment. Furthermore, the constant evolution of ICs Integration Density causes an increment in the MCUs span. MCU solving could be addressed through Multiple Error Correction (MEC) Codes, as explained in Chapter 2, though they are not feasible from the practical point of view due to the computing resources involved (time and logic area) are exponentially increased with the simultaneous bits number to be corrected. Single Error Correction (SEC) Codes are suitable from the computing resources point of view, although the obviously are not able to solve simultaneous bits errors. However, if a subset of bits can be suitably extracted from the whole set of bits in such a way that no-more-than-one-bit-error is guaranteed to be found in the subset, then SEC Codes could be applied to each bits subset. This leads to the Interleaving concept, which is the procedure that allows defining a suitable subset of bits. Whereas MEC Codes are not feasible from the practical point of view to its exponential resources increase with the MCU size, Interleaving-SEC Codes solution allows an affordable solution due to its just linear increase of the operation time with MCU size. A suitable subset of bits is called a “virtual frame” in this work. Virtual frames generation from interleaving concept is explained in Section 4.1.

On the other hand, there are some practical critical applications in which a failure may be defined as catastrophic, which means that it should not happened under any circumstance, frequently due to human life is involved. This leads to the idea of designing Fault Tolerant Systems, in which the system is guaranteed to keep an appropriate response even under a failure condition. These issues are typical, for example, in aviation applications, where failures in electronics under an increased radiated environment may take place.

Fault-Tolerant systems are typically based on redundancy concept for storing and retrieving healthy information, for example, with a Triple Modular Redundancy (TMR) scheme. The main issue with redundancy is design oversizing, since a simple voting system is applied with no Error Correction Codes involved. This Thesis proposes a new hybrid architecture that takes advantage of the optimized performance of Error Correction Codes from the oversize point of view, with extremely compressed redundant information that guarantees the information integrity even if an MCU takes place in the scrubbing function itself. Section 4.2 explains the proposed approach.

4.1 VIRTUAL-FRAMED INTERLEAVING FOR MCU CORRECTION

MCU means that a set of neighbor bits have been simultaneously upset due to a neutron radiation event. Initially, single-error correcting techniques are not enough since more than one cell might be affected at the same time. However, techniques such as the Interleaving Distance (ID) approach enable overcoming it [35], [36]. The interleaving approach of this work mainly consists in reordering the real (physical) bits distribution into a virtual suitable subset of bits (virtual-frame) in such a way that no-more-than-one-bit may be affected by radiation when scrubbing each virtual frame. If the previous condition is fulfilled, single-error correcting techniques can be applied and, therefore, there is a significant performance improvement due to the light-weight SEC codes used.

Typically, MCU correction implies a complex EDAC code that increases the process exponentially in terms of required resources. Interleaving Distance (ID) approach allows splitting the memory into frames to be scrubbed independently. This increases the resources linearly instead of exponentially, [6], [7]. Moreover, whereas detection (very light process based on parity) could be also performed in every frame, correction (still light for a single upset) is only carried out in the affected (detected) frames. As a result, this approach means a significant optimization

In order to guarantee that there is no-more-than-one-bit in each frame, and interleaving distance ID has to be defined depending on the maximum MCU span. Each frame has to be composed of bits which are physically separated at least the MCU span itself. For instance, if a maximum MCU span of 5 bits is currently considered according to experience (Chapter 2), the minimum interleaving distance to be considered should be also 5 in order to split each error bit involved in the MCU into different frames. The interleaving distance has to be applied in both rows and columns in memory.

To clarify this, a simplified memory of 17x8 bits is considered as example to show these concepts, in which a 5 bits MCU has been pointed in red (Fig. 4.1). The MCU span can be considered as 4 in Fig.4.1 since the maximum consecutive affected bits in either a row or column (column in this case) is 4. Therefore, an ID of 4 is chosen in this case as shown in Figure 4.2.

Having this ID value for example purposes, the memory has been divided into 16 frames keeping the same ID distance, 4, in both directions.

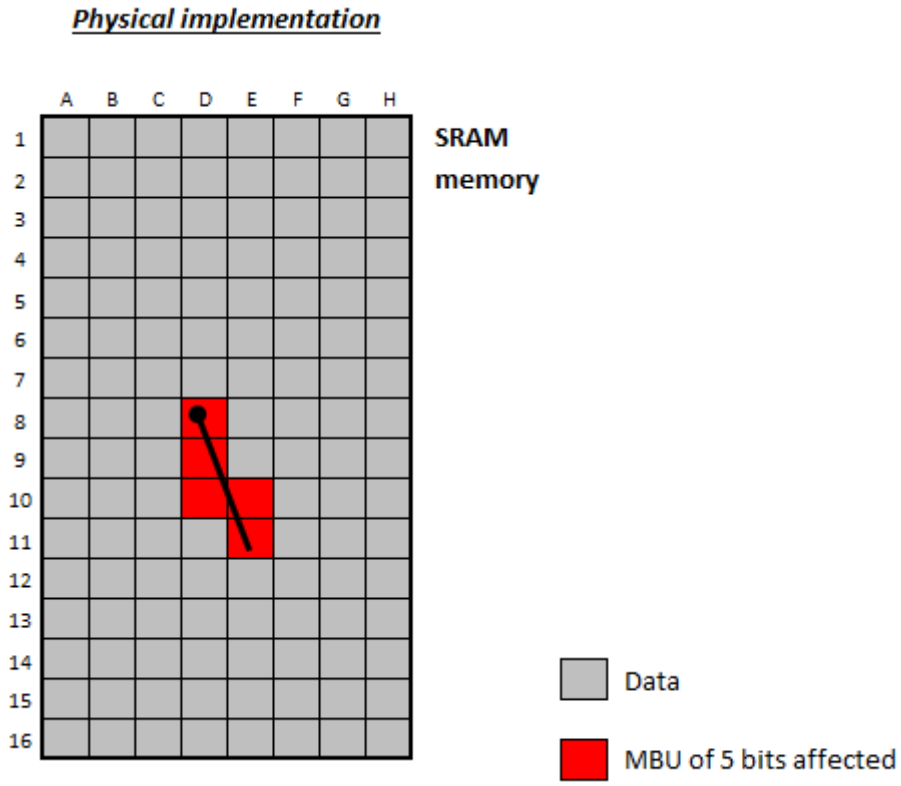


Figure. 4.1. 17 x 8 bit memory divided in 16 frames. Bits affected by MCU are marked in red

As a result:

- The number of virtual frames to be considered depends on the ID distance, 4 in the example. Since the ID distance is applied to both rows and columns, there are 4^2 bits that must belong to a different frame. That means a maximum correction capability of 4x4 bit MCU.

- In a bigger memory, the number of virtual frames would remain the same, since it only depends on the ID. On the other hand, the number of bits included in each virtual frame increases accordingly. For instance, if the memory size is increased from 64 bit to 256 bit (x4) having ID=4 and 16 frames in both cases, the number of bits per virtual frame increases from 4 to 16 (x4).

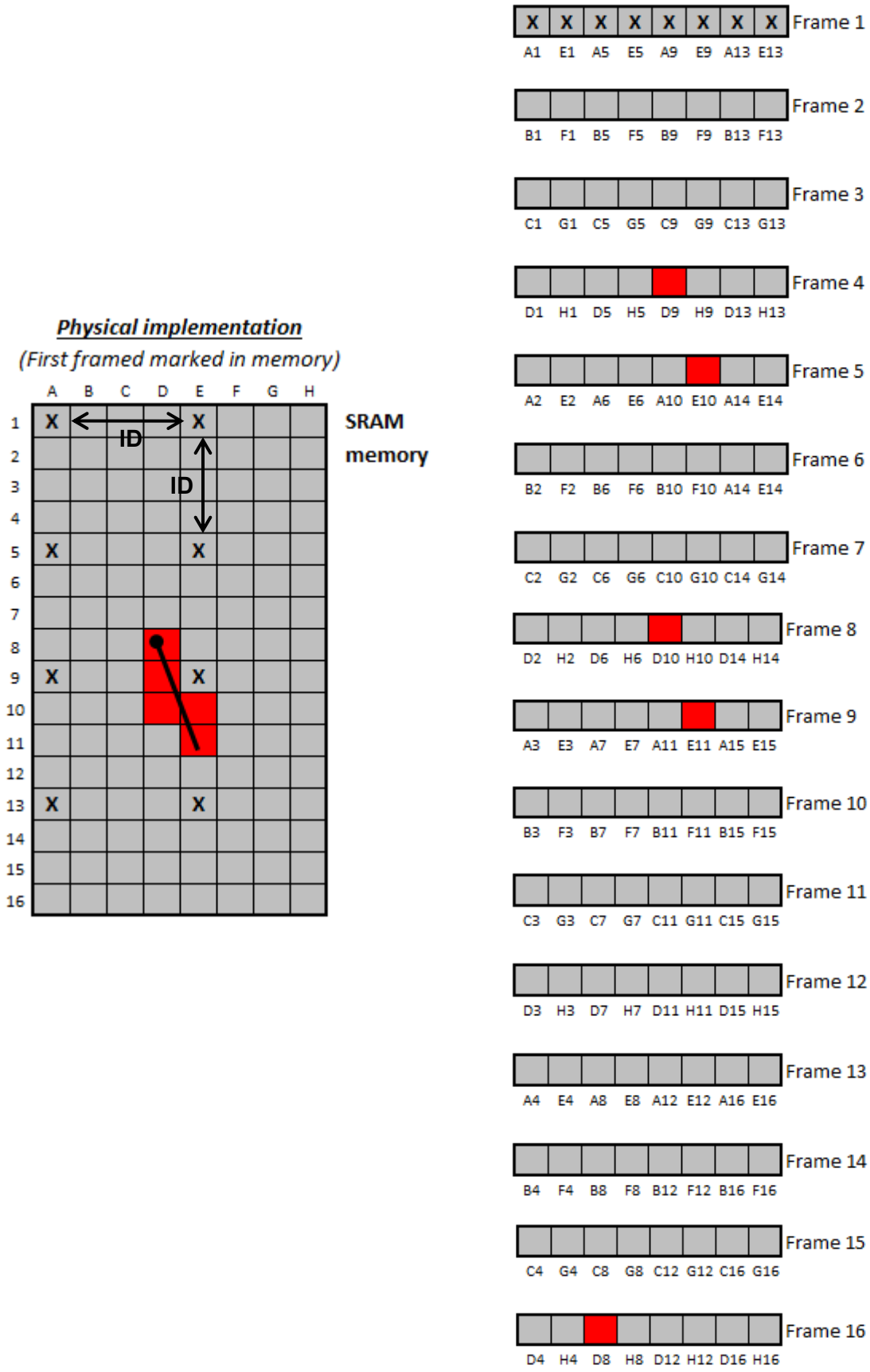


Figure. 4.2. Data bits of Frames affected by MCU event of Fig. 4.1. ID distance = 4

As shown in Figure 4.2, thanks to this suitable distribution of the whole memory matrix into frames, a 5-bit **MCU event leads to a SEU problem in 5 different frames, with only one error per frame.**

This framed approach works successfully because the MCU event example in Fig.4.1 (3-bit span range with 5 bits as total number of bits affected) has a span lower than or equal to the distance, ID, previously set as 4 bits. Furthermore, the total number of bits affected by MCU event is also lower than the square of distance, ID^2, which is 16 bits. This framed architecture allows converting a multiple bit upset problem into a single error one, SEU, distributing each upset bit into a different frame.

According to previous description of minimum distance for the framed architecture, the Interleaving Distance, ID, should be enough to cover the worst MCU pattern from the bit span point of view. According to Chapter 2, the maximum number of bits affected in an MCU event in the current technology state is not higher than 5 bits. This estimated span is expected to change in the future according to different factors such as the device integration density, supply voltages, radiation composition, etc.

Figure 4.3 shows other example based on same memory with a 3-bit MCU event. In this case, the required ID distance would be 2 according with its bit span, which means a total frames number of $2^2 = 4$. As a result (Figure 4.3), a 3-bit MCU event affects to 3 different frames with only one error per frame**,** bit E11 in Frame 1, bit E10 in Frame 3 and bit D10 in Frame 4.

Figure 4.4 represents another example based on same memory with a different MCU, of 5-bits size and 3-bits span. If the ID distance is kept respect Fig 4.3 (ID = 2), then Frame 4 would include 2 upset bits. In this case, the MCU size exceeds the maximum error correction capability and, as a result, the architecture cannot guarantee the SEU approach in all possible MCU cases. The condition to be fulfilled to cover all possible cases is that **the ID value must be at least the maximum estimated span of all the considered MCU events**.

This ID defines the number of frames in which the memory is divided, ID^2 frames. From this definition, the procedure to evaluate the status of the memory requires the scrubbing of the complete memory sequentially masking the data into ID^2 frames.

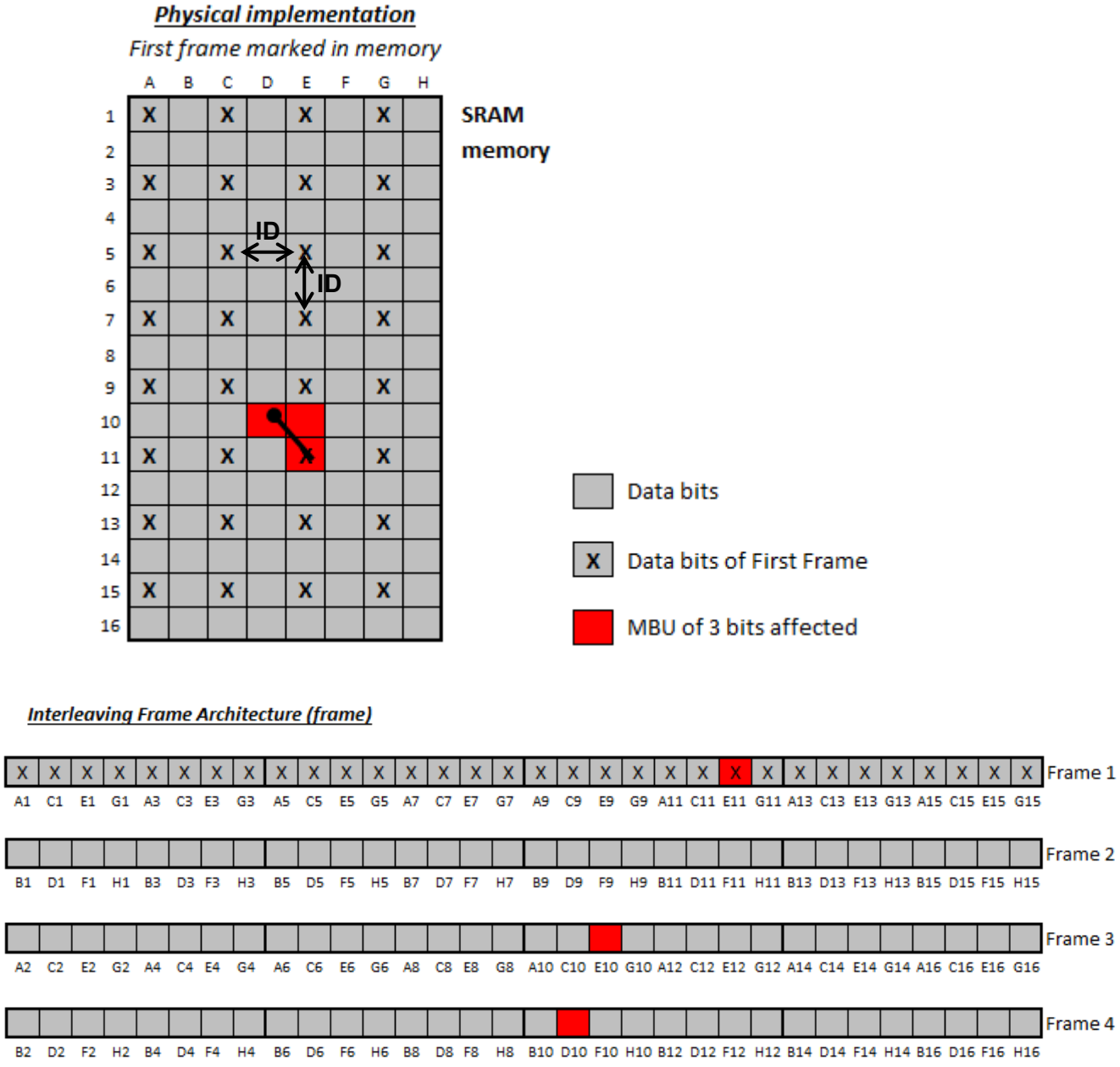


Figure 4.3. Bits of Frames affected by MCU event. ID distance = 2

This procedure is implemented through an algorithm which computes only one frame every iteration cycle. Figure 4.5 represents the operational flux algorithm according to the steps commented before. This procedure is the same as the one used in Figure 3.5 for SEU correction. This algorithm is expanded with the interleaving distance technique by introducing the frame masking every cycle. Thus, same operations performed for single error detection and correction at Figure 3.5 are still valid within the frame masking for the MCU event separated in frames.

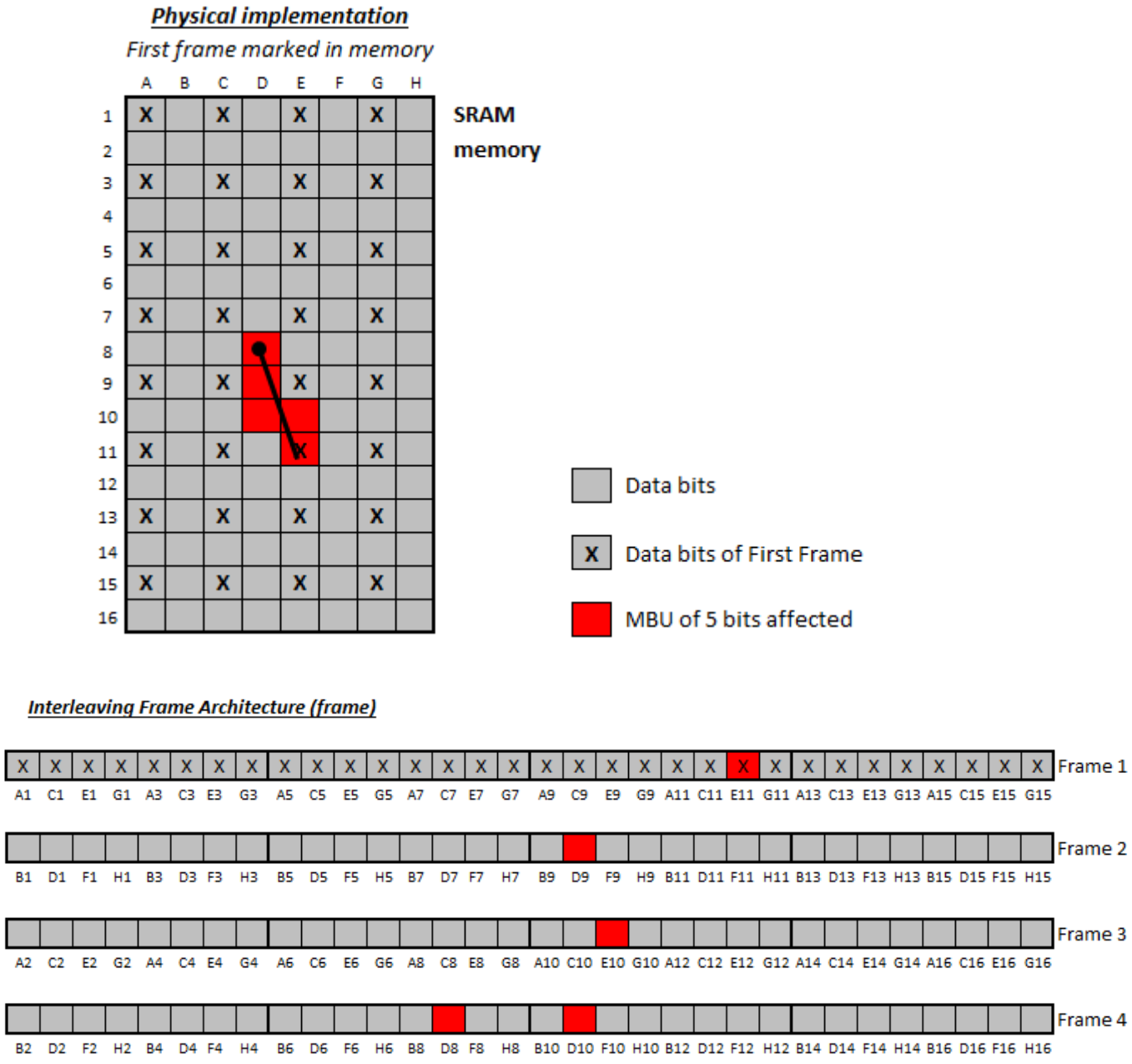


Figure 4.4. Bits of Frames affected by MCU event. ID distance = 2

In case of no errors injected in frame, there are just two computations that are performed every iteration cycle in the principal operation line (same as Fig 3.5), the Checksum validation by the SEC (Single Error Correction) functionality with HSBs as reference and then the Parity of Data.

The introduction of a parity bit in the checksum level in each frame significantly reduces the timing spent in the scrubbing function. This is because this architecture will only launch the correction procedure, which implies much

more time than the detection procedure, parity bit, just when a failure is previously detected in the computed frame in each cycle.

Only in case that parity detects an error in any of these two steps, SEC is launched in order to correct it. Since no-more-than-one error is guaranteed to take place in each frame, there is no need of using Multiple Correction codes and just a Single Correction one is required. In order to achieve it, the ID value has to be higher than the expected MCU span in order to reduce a MCU event problem as just a SEU one at each frame. Thus, a total frames number of ID^2 are independently scrubbed under the sequential procedure of Fig. 4.5.

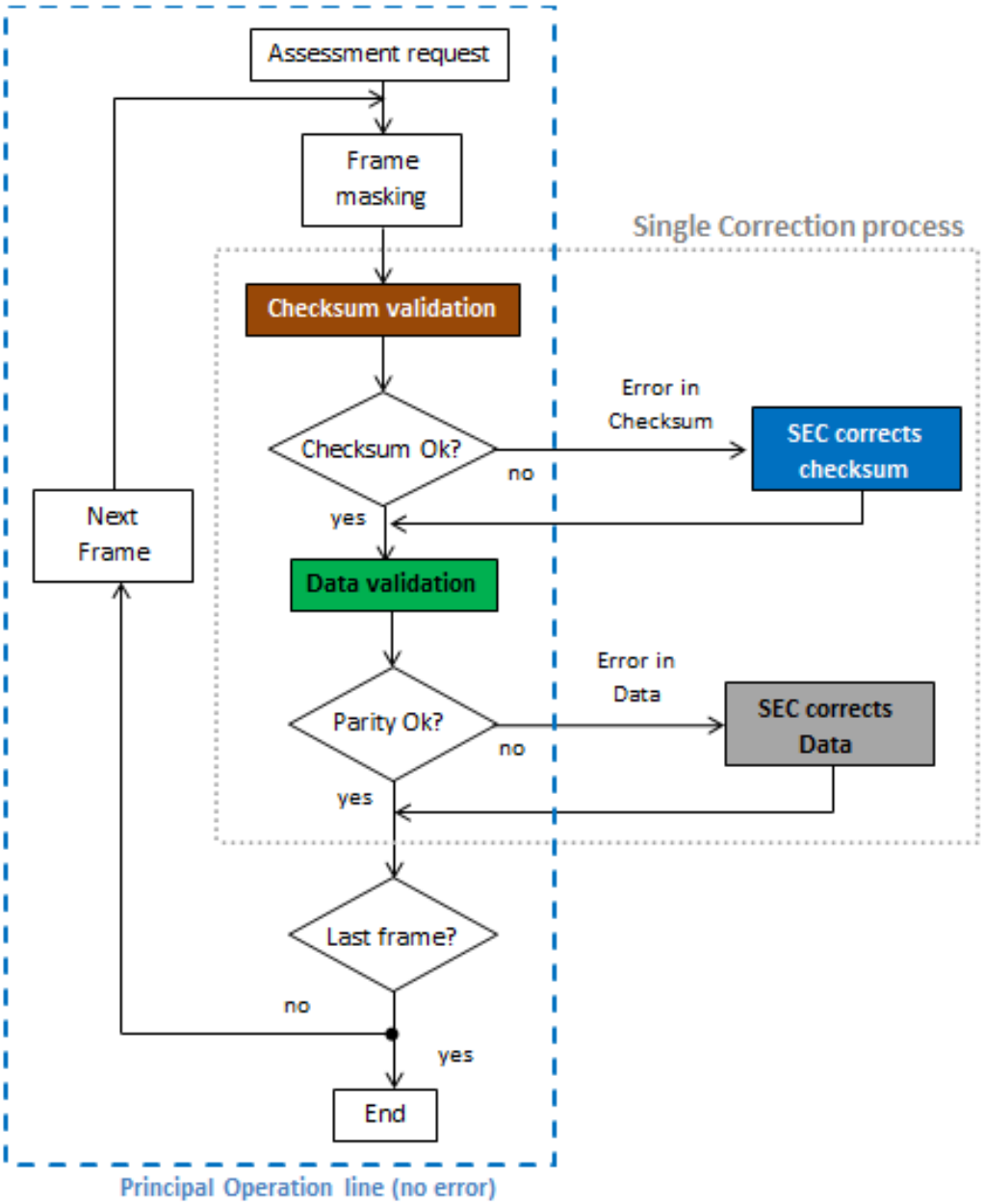


Figure 4.5. Algorithm established to mask the memory for scrubbing

The main magnitudes to be considered are pointed at table 4.1. The correspondent expressions are presented in equations 4.1. to 4.9.

$$FN = \mathrm{ID}^2 \quad \text{Eq. 4.1}$$

$$ID \geq MCUspan \quad \text{Eq. 4.2}$$

$$NF \geq MCUbits \quad \text{Eq. 4.3}$$

$$FB = \frac{DS}{NF} = \frac{m \cdot n}{NF} \quad \text{Eq. 4.4}$$

$$2^{CHB} - 1 \geq FB \quad \text{Eq. 4.5}$$

$$Overhead\_CH = (CHB + PB) \cdot FN \quad \text{Eq. 4.7}$$

$$\mathrm{MS} = \mathrm{DS} + \mathrm{Overhead\_CH} \quad \text{Eq. 4.8}$$

$$2^{\mathrm{HSB}} - 1 \geq \mathrm{CHB} + \mathrm{PB} \quad \text{Eq. 4.9}$$

$$Overhead\_HSB = HSB \cdot FN \quad \text{Eq. 4.10}$$

ID	**Interleaving Distance.** This interleaving defines the maximum capability to correct pattern of MCU. ID shall be selected with a value higher than the maximum expected span of MCU in target memory
FN	**Number of Frames** in architecture
MCUspan	Maximum consecutive affected bits in either a row or column in a memory
MCUbits	Total expected number of affected bits by a single impact.
DS	**Data Size.** m is the number of words of memory (rows in the example) n is the number of bits per word (columns in the example)
FB	**Frame Bits.** Number of data bits that compose each frame
CHB	**Checksum Bits.** In a SEC code, the checksum is the bit set required to protect the data of a frame. If X is the checksum size (number of bits), up to 2^X-1 data bits can be retrieved, if no-more-than-one error takes place.
PB	**Parity Bits.** In a SEC code, one parity bit is included per frame in order to detect errors in data of a frame.
Overhead_CH	Total number of redundant bits required in architecture to protect the data of a memory
MS	**Memory Size** is the total memory bits number, which includes the Data bits and the Overhead_CH bits.
HSB	The HSB is the bit set required to protect the Checksum and parity bits of a frame. If X is the HSB size, up to 2^X-1 data bits can be retrieved, if no-more-than-one error takes place.
Overhead_HSB	Total number of redundant HSB required in architecture to protect the Checksum and parity bits of a memory

Table 4.1 Summary of concepts for HSB Framed architecture

In order to explain the integration of the architecture with the interleaving approach in standard memories, a couple of numerical examples are presented in Table 4.2 for two different memory sizes.

The few redundant HSB bits required in each memory is calculated also in Table 4.2 where only 4 HSB per framed is required in a 128 Kbits memory, a total of 0,2% of the complete memory. For 1 Mbit memory, only one more bit is required per frame, 5 HSB per framed, being a total of 0,00003 % of the complete memory.

Variable	Concept	128 Kbits	1 Mbit
ID	Interleaving Distance	8 bits	8 bits
FN	Frames number	64 frames	64 frames
MCUspan	MCU span bits	5 bits	5 bits
MCUbits	Maximum MCU bits	12 bits	12 bits
DS	Data size	4.088 words of 32 bits each (130.816 data bits)	33.554.432 words of 32 bits each (2^{30} data bits)
FB	Frame bits	2.044 bits per frame	524.288 bits per frame
CHB	Checksum bits	11 checksum bits per frame	20 checksum bits per frame
PB	Parity bits	1 parity bit per frame	1 parity bit per frame
Overhead CH	Number redundant bits required to protect DS	768 redundant bits in checksum (0,6 %)	1.344 redundant bits in checksum (0,0001 %)
MS	Memory Size	24 extra words 4112 words of 32 bits each (131.584 memory bits)	42 extra words A total of 33.554.474 words of 32 bits each (2^{30} memory bits)
HSB	Hardwired Seed Bits	**4 HSB per frame**	**5 HSB per frame**
Overhead HSB	Number redundant HSB required to protect MS	256 HSB **(0,2 %)**	320 HSB **(0,00003 %)**

Table 4.2 Data obtained for main parameters of HSB Framed architecture for different memories

The bits which are included in the first frame are pointed in Figure 4.6. They have been obtained through a bits masking of 8-bit ID separation. According to the selected 8-bit distance in both dimensions, a total of 64 frames (Eq 4.1) are needed to cover whole memory matrix.

As a reference, some Xilinx FPGA memories [31], [32] are an actual example of the interleaving concept application. They apply it in just one direction, which means that they are divided into several CLB columns (Configurable Logic Block) that contain up to 48 frames per CLB. Each frame is protected with a Multi-bit ECC (Error Correcting Code) able to correct up to 4 bits per frame. Particularly, Xilinx memories define an ID of 2 bits, which means that in the end, up to 8 bits per frame can be simultaneously corrected.

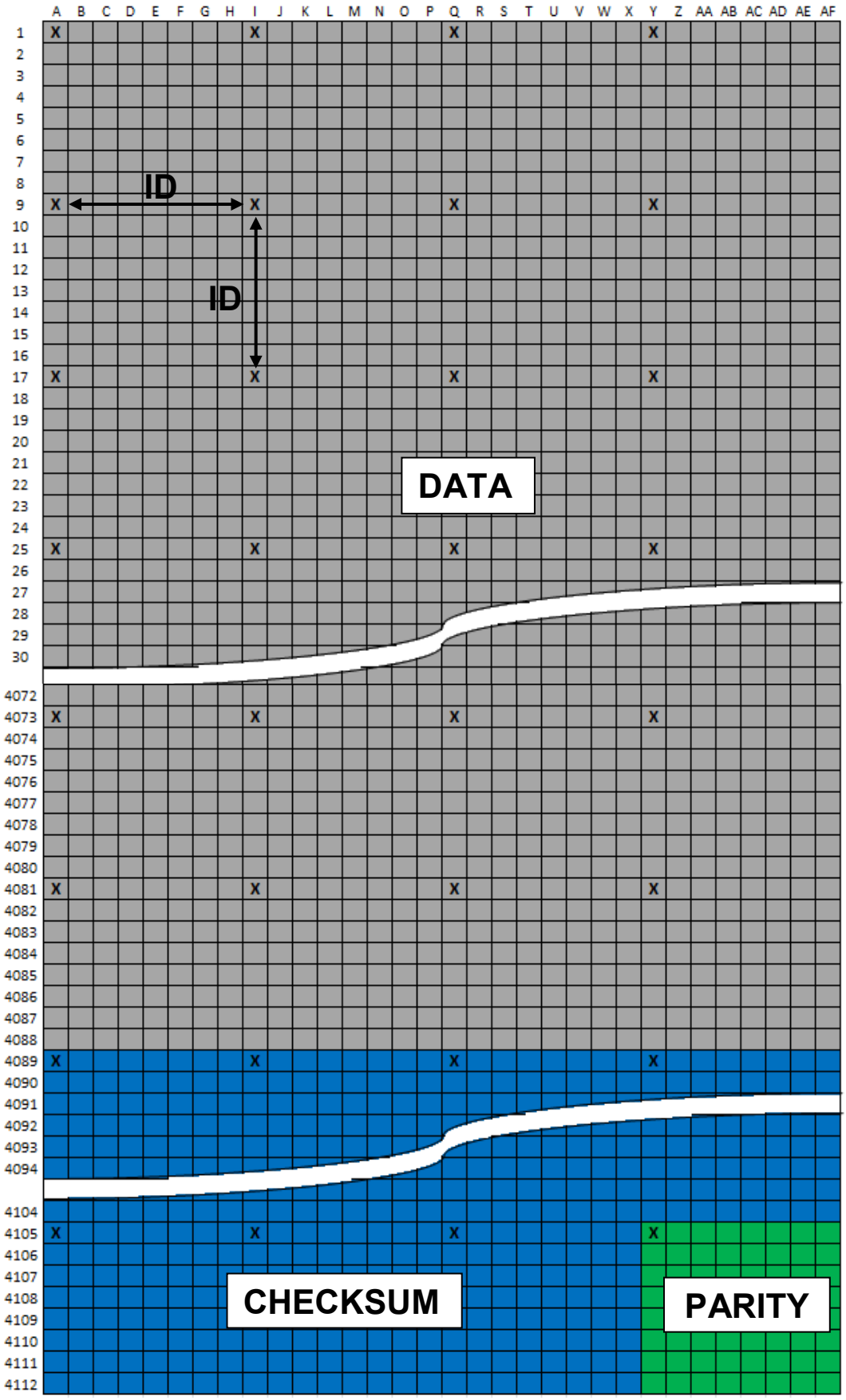


Figure 4.6. Architecture distribution in SRAM-based FPGA with ID equal to 8. Grey bits correspond to data level bits, blue bits with Checkword level bits and green bits corresponds with the parity bits.

4.2 ARCHITECTURE FOR FAULT TOLERANT SYSTEMS

In the Aerospace Industry, Fault Tolerant Architecture requires that the failure probability has to be no greater than the requested failure rate. Fault Tolerant (FT) systems allow continuous proper operation even under a failure of any of its components. This means that the system keeps its functionalities even when an event takes place at the scrubbing function itself. TMR is currently the most used technique to mitigate failures in safety critical avionics systems, minimizing the failure probability.

As exposed in Chapter 2, the classical SEU mitigation methods to protect a fault tolerant system are based on system triplication. Each triplicated copy performs the same process and the three results are combined under a voting approach to produce a single output. In case of a bit upset in one of the copies, there is still a right memory reading due to the error masking supported by the other two healthy copies, and eventually no failure takes place in the system. Furthermore, it would also require an additional reconfiguration-based technique implementation (additional resources) for the error correction in order to avoid an error accumulation in memory.

Other simpler method for SEU correction is to omit readback and SEUs detection by simply partially reloading the entire damaged Frame segment at a chosen frequency. This is called "scrubbing." Scrubbing requires that the configuration logic to be in "write mode" for a greater percentage of time, with the corresponding time involved. Furthermore, a non-susceptible (no affected by Radiation effects, SEU or MCU) hard copy of the complete memory, , is required to be implemented as the scrubbing data source, which means a significant overhead data. Furthermore, this non-susceptible memory normally presents high reading time, which has a significant impact in the operation time.

The Thesis proposed Architecture (which algorithm is presented in Fig. 4.7) aims to optimize the previous solutions from the data overhead and the computing time points of view, and yet allowing meeting FT requirements. It is based on the concepts previously explained in Chapters 3 and 4, adding just a SEC copy in the memory at Data level (Fig. 3.1), under a what is called a DMR-SEC approach (Double Modular Redundancy for Single Event Correction). This means that just the SEC function is duplicated instead of the whole memory triplication.

The algorithm represented in Fig. 4.7 is similar to the algorithms exposed previously in Chapter 3 and 4.1 although it includes the DMR SEC functionality for Fault-Tolerant system.

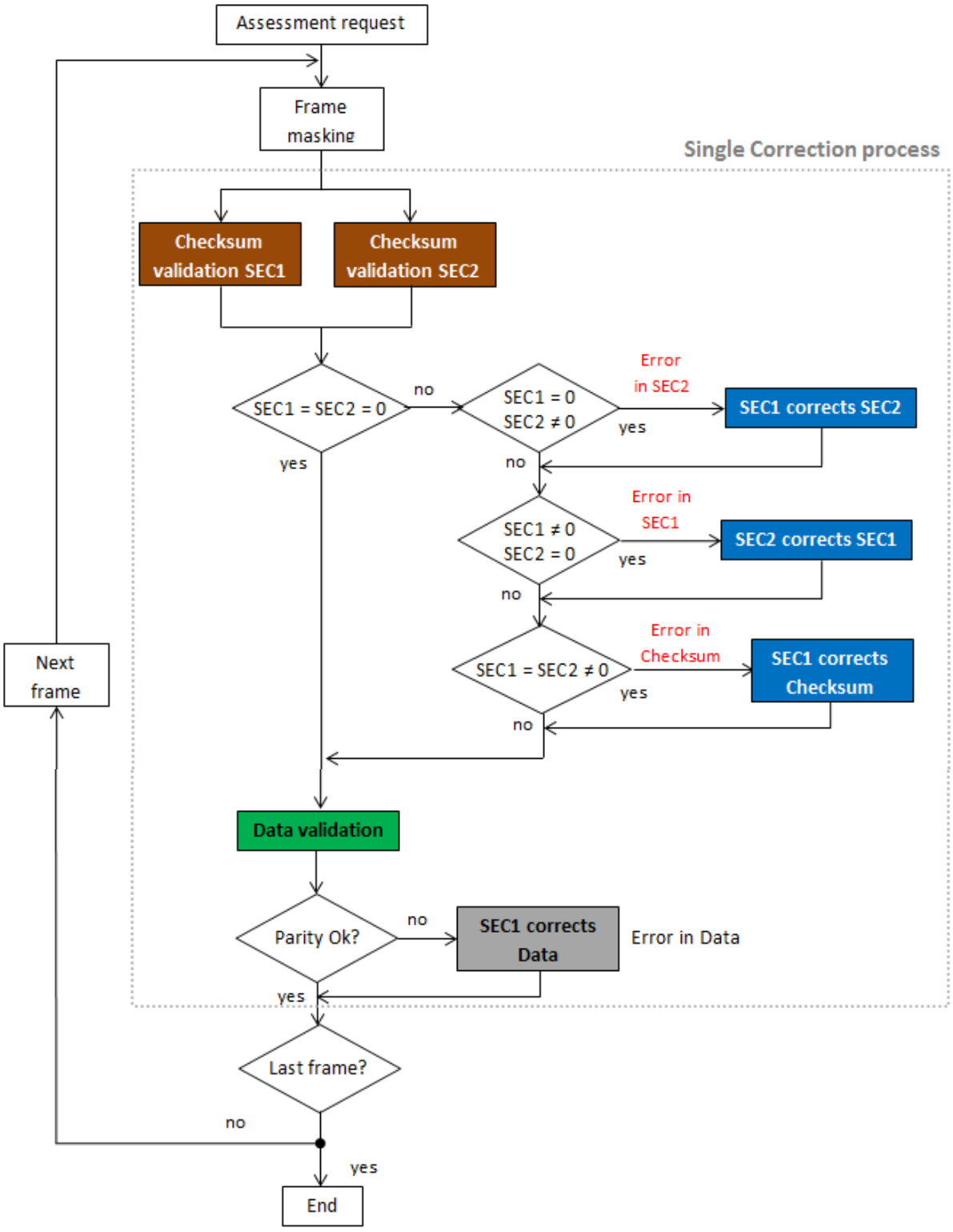


Figure 4.7. Algorithm adapted for Fault-tolerant system

The algorithm, Fig 4.7, is executed each iteration in only one frame. In case of no errors injected in memory, the Principal Operational Line is similar as the presented in Fig. 3.5 and Fig. 4.5, apart from the fact the checksum is checked twice (by SEC1 and SEC2 respectively) instead of just once.

The Algorithm is an iterative process. Each iteration starts masking a new frame from memory to compose the three levels of the architecture: HSB, checksum and parity, and data bits. A counter is requested to keep the sequential procedure to complete memory after ID^2 cycles.

The algorithm structure is composed of 2 Steps, each of them presenting different Scenarios (Table 4.3.).

Step	Scenario	Reference	SEC1	SEC2	Action
1	1	HSB	No error	No error	No action
1	2	HSB	CS error	CS error	CS correction
1	3	HSB	No error	CS error	SEC2 correction
1	4	HSB	CS error	No error	SEC1 correction
2	1	Parity	Parity error	-	Correction launching
2	2	Parity	No error	-	No action
2	3	Checksum	Data Error	-	Data correction

Table 4.3 Possible scenarios oriented to tolerant fault requirements

STEP 1. Once frame is composed, both SEC1 and SEC2 are computed from the permanent reference, HSB, in order to validate the Checksum. The main idea is based on interleaving, which was applied to mask a Frame to guarantee that no more than one error can take place at the same time in it. Therefore, if both functions agree, they both have to be right. If they do not agree, one of them has an error. There are The possible Scenarios to consider under Step 1, are:

Scenario 1.1: If the results of both functions are zero, (SEC1 = 0 AND SEC2 = 0), that means that both SEC function agrees that there is no error in the Checksum level. Since this result is obtained at a Frame in which no more than one error is guaranteed, the only possibility is that both SEC1 and SEC2 and also the Checksum level are cleared from errors. This operation validates the use of any of the SEC functions and also the Checksum data for following operations of procedure of Step 2 with no further action.

Scenario 1.2: both functions agrees in identifying an error in checksum, obtained results in both functions are different than zero, (SEC1 = SEC2 ≠ 0). Again, if they agree, they are right and cleared, whereas there is an error at checksum level. Correction procedure is launched in order to correct the

identified error in Checksum bits. SEC1 is defined in Fig.4.7. as the default function to perform this Scenario, although SEC2 is also valid in this case.

Scenarios 1.3 and 1.4: If different results are obtained between scrubbing functionalities, (SEC1=0 AND SEC2≠0 or vice versa), that means that an error is found in SEC2 (or vice versa) and, therefore, Checksum have no errors since no more than one error is guaranteed. The function with the result different than zero is the function that is in error while the other function is the healthy function with a result equal to zero, coherent with the Checksum level state. The procedure uses the healthy SEC function to retrieve the damaged one from the HSB.

STEP 2. After Step 1 procedure, SEC1, SEC2 and Checksum level are considered cleared. Parity check is then performed on Data level from Checksum Level through SEC function (SEC1 in Fig. 4.7.). The possible Scenarios in Step 2 are the following:

Scenario 2.1: If parity correlation does not match, an error is detected in data bits of frame computed. Therefore, correction procedure is launched only for this frame.

Scenario 2.2: If parity matches, no error is detected in data of this frame. Therefore, next frame is then masked.

Scenario 2.3: If correction procedure is launched, the procedure computes the data through the Systematic linear block code, explained in Chapter 4.4, and correlates its results against the checksum bits. The syndrome of this operation reveals the bit in data that is in fault. Finally, the error is corrected by flipping the current state.

If an error were found in any SEC functionality, Scenarios 1.3 or 1.4 of Table 4.3, the error is naturally corrected at Step 2 since the right SEC function to be used in Step 2 was identified at Step 1 and the damaged SEC function is part of the Data Level.

This algorithm reveals the Fault Tolerant capability under a self-protection approach based on HSB as the root of a Multilevel Framed HSB Architecture. It only needs to duplicate its scrubbing function (SEC1 and SEC2). There is an additional requirement to rightly implement the interleaving approach in order to guarantee that at least one of the SEC functions is in healthy condition, which is to allocate them at least at a ID distance away or within the memory. This requirement is very easy to fulfill taking into account an ID = 8 bits in a current memory size, but still it has to be considered. This requirement is illustrated in

Figures 4.8 and 4.9. In order to avoid a common failure that introduce a double errors as explained in Step 1 and Scenario 5, Figure 4.8.a represents the scenario in which both SEC functionalities could be affected by only one neutron impact. This situation shall be avoided by using the configuration shown in Figure 4.8.b.

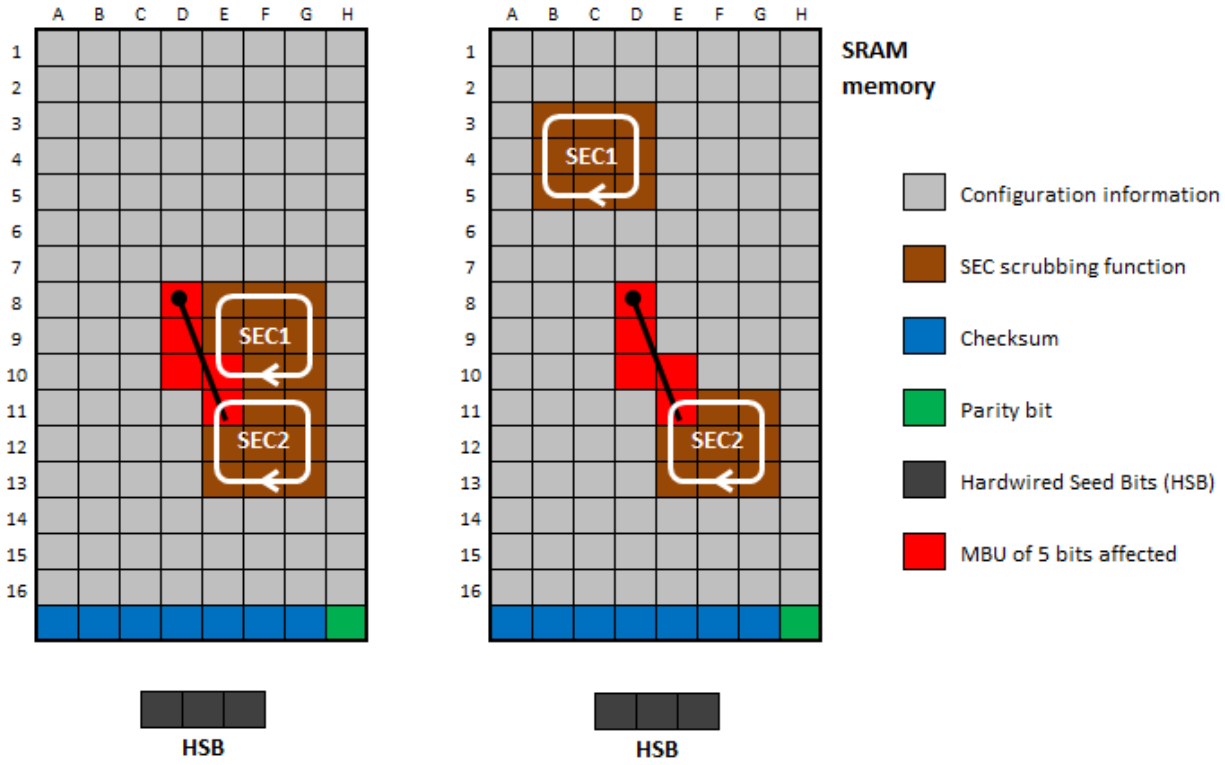


Figure 4.8. HSB architecture distribution where both duplicated scrubbing functionalities are placed in Data level. MCU event could affect to both SECs at same time (left). This situation shall be avoided. MCU event only affects to SEC2 but no to SEC1 (right).

As a summary of conditions that shall be met in the hardware design for this architecture to guarantee the operation without double errors:

- Space condition: The twin scrubbing functions, SEC1 and SEC2, and the Checkword must be placed physically apart from each other, at least right at the maximum expected MCU span.

- Time condition: the time involved in the whole process for a frame scrubbing must be low enough to neglect the probability of experiencing two radiation events within the same iteration.

Under both conditions, no more than one of these 3 key blocks (twin SECs and Checkword) can be considered corrupted at the same time.

4.3 IMPLEMENTATION ISSUES

ISSUE 4: Fault Tolerant Probability Threshold

All actual systems are susceptible to failure. Therefore, the Fault Tolerant definition must be stablished in terms of a failure probability threshold. Actually, in the standards stablished in the aerospace industry (ARP4754, ARP4754/A, RTCA DO254 and RTCA DO178) the severity of a system is defined in terms of DAL (Design Assurance Level). The most severe scenario is named as a catastrophic event, and the systems and equipments that may produce it in case of failure must be certified as DAL-A. According to standards, the failure probability of DAL-A systems must be no greater than $1 \cdot 10^{-9}$ 1/hr. This would means, as a rough example, that just one catastrophic failure would take place in the whole life-span of a complete fleet (1.000 aircrafts).

The proposed Architecture is based on the capability of having no more than one event per scrubbing cycle, in order to guarantee that no more than one bit is affected in a frame, and that Fault Tolerant requirements are fulfilled under a DMR-SEC approach (Double Modular Redundancy with a Single Event Correction). Even having a double (or even more) radiation events in the same scrubbing cycle, the proposed Architecture could retrieve them if the impacts take place in different frames.

The issue to consider in this case is:

- How to calculate the real failure probability taking into account all possible scenarios and which are the design degrees of freedom of the proposed Architecture that allows reaching the Fault Tolerant Threshold? This Issue is discussed in Chapter 5.

4.4 DETAILED CODING IMPLEMENTATION

4.4.1 CHECKSUM CODING FOR SEU CORRECTION

The proposed solution is based on a **Reconfiguration-based solution.** This tries to restore the original values in memory as soon as possible after SEE event. It is a **Single Error detection and Correction code (SEC)** based on checksum codification that is a modular arithmetic sum of bits of message codewords of a fixed word length, called **systematic linear codification**, [5] and [6].

This SEC is performed as background functionality to restore the upsets as soon as possible to the original values in memory after SEE event. It is performed periodically depending on the criticism of functionality and the SEU rate predictions for a given application or mission.

Systematic linear block code is used for the purpose of detecting errors but also for correcting errors in certain cases. The redundant checksum bits obtained through this codification allows detecting and correcting a limited number of errors that may occur anywhere in the data bits. By far, main advantage of this kind of linear codes is the light-weighted correction process because the calculation of each checksum bit is based on parity computations. The checksum word has also the property that the original information data bits also appear on the checksum word unchanged with some parity bits added, called “systematic code”.

D1	D2	D3	D4	D5	D6	D7	C1	C2	C3
X		X		X		X	X		
	X	X			X	X		X	
			X	X	X	X			X

1	0	1	0	0	1	1	1	1	0
1	1	1	0	0	1	1	1	0	0
1	0	1	0	1	1	1	0	1	1

Figure 4.10. a) Checksum concept for a 7-bit data and 3-bit checksum.
b) Example of a 7-bit data and checksum

where:

$$C1 = D1 \text{ xor } D3 \text{ xor } D5 \text{ xor } D7,$$

$$C2 = D2 \text{ xor } D3 \text{ xor } D6 \text{ xor } D7 \text{ and}$$

$$C3 = D4 \text{ xor } D5 \text{ xor } D6 \text{ xor } D7$$

Figure 4.10 shows and example of the checksum coding process. To check the integrity of a message, the receiver computes the exclusive or (XOR) of all its words including the checksum, following the codification pattern fixed. If the result is not a word with n zeros, the receiver knows a transmission error occurred. With this checksum, any transmission error which flips a single bit of the message, or an odd number of bits, will be detected as an incorrect checksum. However, an error which affects two bits will not be detected if those bits lie at the same position of one checksum bit.

The checksum bits shown in Figure 4.10, C1, C2 and C3, are obtained by means the XOR computation of data bits, while data bits, D1 to D7, included in a unique set and determined by the binary form of its bit position. General Checksum bits are defined as:

- Checksum bit C1 covers all bit positions which have the least significant bit set to 1: bit 1, 3, 5, 7, 9, etc. The computation is always assigned as the Less Significant Bit, LSB. The weight value for syndrome computation for an error is equal to 2^0.

- Checksum bit C2 covers all bit positions which have the second least significant bit set to 1: bit 2, 3, 6, 7, 10, 11, etc. being defined as 2^1 for syndrome computation.

- Checksum bit C3 covers all bit positions which have the third least significant bit set to 1: bits 4–7, 12–15, 20–23, etc. being in this example, the Most Significant Bit, MSB, for syndrome computation, 2^2, when error is detected.

- Checksum bit C4 covers all bit positions which have the fourth least significant bit set to 1: bits 8–15, 24–31, 40–47, etc. would be defined as 2^3 for syndrome computation.

As shown in Figure 4.11, a combination of m different checksum bits can protect data bits up to 2^m-1. Total length (data plus checksum) is obtained by summing the data bits plus checksum, 2^m+m-1. As m varies, all the possible Checksum with maximum capability can be obtained. In previous examples, 3

checksum bits cover up to 7 data bits while 5 checksum bits would cover up to 31 data bits. Any intermediate combination could be also constructed although it is not optimized. For example, a frame of 22 data bits could be constructed and it is protected by 5 checksum bits.

	m	3	4	5	6	7	8	9
Data bits	2^m-1	7	15	31	63	127	255	511
Checksum bits	2^m+(m-1)	10	19	36	69	134	263	520

Figure 4.11. Different Bit ranges of Checksum up to m=9

The redundant checksum bits obtained through this codification allows detecting/correcting a limited number of errors that may occur anywhere in the data bits, **not in the own redundant bits**. If an error occurred in these redundant bits, new checkword will be wrongly understood as a failure in data. Errors in redundant bits cannot be corrected. For that reason, multilevel architecture is intended to protect only the redundant checksum bits because they protect these redundant bits.

This correction capability of checkword is shown in Figure 4.12. In those binary combinations, some error cases are included in same reference word, “1010011” whose checksum is “110”. The complete binary word is defined as “1010011110”. Some SEU errors are included as an example in different parts of the complete word to analyze its effects.

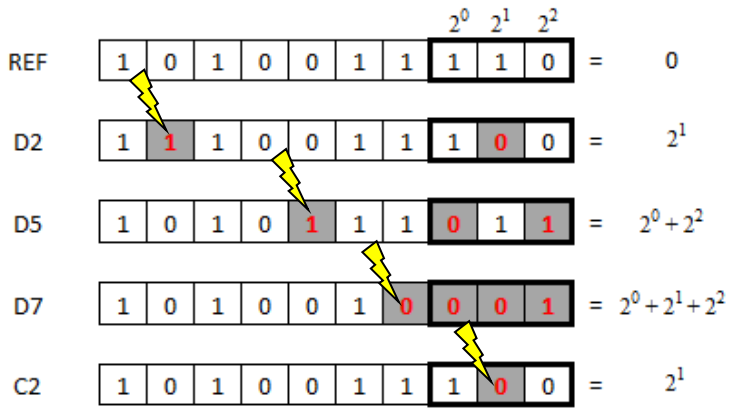


Figure 4.12 represents some examples of error injected in different parts of the computed referenced word:

- Ref case: Having the original bitstream set as “1010011”, whose parity checksum is obtained as “110”, the complete word is defined as “1010011110”. This checksum bitstream “110” will be used as a reference gold value to compare in any scrubbing cycle.

- D2 case: Having a failure in second bit of bitstream, D2, "1**1**10011", it causes a new recomputed checksum that is "100". It does no match with the saved gold checksum "110" and has only one different bit when compared as known C2. According to weighted value of every checksum bit in failure, as only C2 is in failure, it weighted value corresponds to 2^1, that gives a syndrome in bit 2 as bit upset.

- D5 case: Having a failure in fifth bit of bitstream, D5, "1010**1**11", it causes a new recomputed checksum that is "011" that differs from the gold value saved. It does no match with checksum "110" and has two different bits when compared as known C1 and C3. According to weighted value of every checksum bit in failure, as C1 is in failure, it weighted value corresponds to 2^0, and as C3 is in failure, it weighted value corresponds to 2^2, that gives a syndrome in bit $2^0 + 2^2 = 5$ as bit upset.

- D7 case: Having a failure in seventh bit of bitstream, D7, "101001**0**", it causes a new recomputed checksum that is "001" that differs from the gold value saved. It does no match with checksum "110" and has three different bits when compared as known C1, C2 and C3. According to weighted value of every checksum bit in failure, as C1 is in failure, it weighted value corresponds to 2^0, as C2 is in failure, it weighted value corresponds to 2^1 and as C3 is in failure, it weighted value corresponds to 2^2, that gives a syndrome in bit $2^0 + 2^1 + 2^2 = 7$ as bit upset.

- In last example, failure only in C2, "1**0**0", it causes a wrong correction of a bit within data bits. In this example, modified checksum, "100" differs from golden value only in C2 that is the bit upset. As C2 is in failure, it weighted value corresponds to 2^1, that gives a syndrome in bit $2^1 = 2$, D2, as bit upset. Therefore, it is the same as a failure detected in second example, failure in D2.

As commented before, checksum can detect errors in whole data including redundant bits but the correction capability for single errors in data bits is only possible if there is no error in checksum bits. Only errors in data bits can be corrected. If the computed checksum for current data matches against the stored value of the previously computed checksum, it is guaranteed that the data has not been accidentally altered or corrupted.

The Generator matrix required for this checksum coding is shown in Figure 4.13. If there are k bits of data, matrix should contain at least k linearly independent codewords to produce linear combinations of two or more

codewords in the set. Easiest way to included k linear independent codewords is to choose the identity matrix as I_K in Generator matrix while A is the representation of linear modulo 2-sum of data bits:

$$G = (I_K \mid A)$$

G =

$$\left(\begin{array}{ccccccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right)$$

Figure 4.13. Generator matrix for the linear block code

The process of encoding is represented in matrix form as:

$$v = u\, G$$

Where u is the information of data bits block, v the complete checksum word, including the redundant bits, and G is the generator matrix. It is important to highlight that generator matrix is a k x n matrix where k is the dimension of the data bits and n is the total length of the checksum word that includes the data bits and the parity checksum. This special form corresponds to a systematic code that contains the k x k identity matrix and k x (n-k) matrix of checksum bits. Thus, same example is obtained by:

$$v = [1\ 0\ 1\ 0\ 0\ 1\ 1]\ G = [1\ 0\ 1\ 0\ 0\ 1\ 1\ \underline{\mathbf{1\ 1\ 0}}]$$

Error detection involves the decision of which bit is in fault by means the analysis of syndrome information. Syndrome is obtained from the parity check matrix composed by the (n-k) x n matrix from generator matrix, G, plus the identity matrix as follows:

$$H = (A^T \mid I_{N-K})$$

H =

$$\left(\begin{array}{ccccccc|ccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right)$$

Figure 4.14. Syndrome matrix for the linear block code

Syndrome: s = H v

$$s = H \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Figure 4.15. No error obtained through syndrome calculation

As commented previously, syndrome result is compared against the original golden value stored in memory to obtain which are the checksum bits weighted that are modified. According to syndrome calculation, if a correct word including the checksum bits is calculated with H matrix, no error is found in syndrome result, having the zero vector.

As an example, D5 case is reproduced again. Bitstream is modified in the way as "1010**1**111110" having a failure in fifth bit of bitstream, D5. When syndrome operation in performed, a syndrome different from zero vector is found, "101", that means that there are two syndrome bits different from original one known as C1 and C3. According to weighted value of every checksum bit in failure, as C1 is in failure, it weighted value corresponds to 2^0, and as C3 is in failure, it weighted value corresponds to 2^2, that gives a syndrome in bit $2^0 + 2^2 = 5$ as bit upset, D5.

As commented before, results obtained by syndrome calculation should be compared against the original golden value stored in memory to obtain which checksum bits are in error. Thanks to the matrix method, after Syndrome matrix calculation, the result obtained gives directly the number of the data bit that is in error to correct it. Once the non-zero vector is obtained after Syndrome matrix computation, each bit should be weighted to obtain the failure bit in bitstream. In example of Figure 4.16, the results obtained from Syndrome matrix computation is "101" that corresponds to $1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 = 5$, being D5 the bit that is in error.

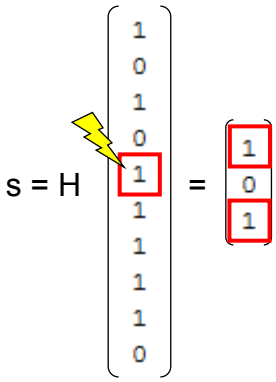


Figure 4.16. Error obtained through syndrome calculation in $2^0 + 2^2 = 5$, D5

Once the data bit indicated by the matrix detection process is corrected, if syndrome is checked again, error should be disappeared and syndrome result should be the zero vector, no error found in bitstream.

Last example included in figure 4.17 is a failure only in C2, it causes a wrong correction of a bit within data-input bit, D2. In this example, modified checksum, "1**0**0" differs from golden value, "110", only in the C2, that is weighted as $2^1 = 2$, bit number 2 of bitstream, D2. Furthermore, it can also be stated by the syndrome matrix calculation. New bitstream, "10100111**0**0", is computed with the Syndrome matrix as shown in Figure 4.17.

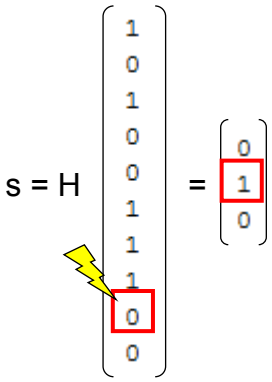


Figure 4.17. Error obtained wrongly through syndrome calculation in $2^1 = 2$, C2

Main advantage of this codification is that redundant checksum bits are obtained through a light-weighted correction process based on parity computations of alternate bits of data-input. Main drawback of this codification is that this codification allows detecting/correcting a limited number of errors that may occur anywhere in the data bits, **not in the own redundant bits**.

In case of MCU event when two or more bits are upset, the checksum coding fails in the detection as shown in Figure 4.18. The results of the syndrome calculation of the bitstream “10**01**011111”, marks “111”. If each bit of the syndrome results obtained is weighted accordingly, it corresponds to $1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 = 7$, that gives a syndrome in bit D7, as bit upset, that is wrongly detected.

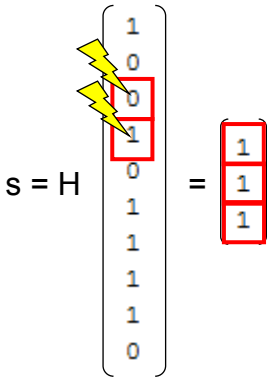


Figure 4.18. Error obtained wrongly through syndrome calculation due to MCU

4.4.2 MULTILEVEL ARCHITECTURE

Checksum code proposed in previous paragraphs is able to correct any single failure included in data bits of bitstream but no anywhere in checksum bits, because it would be wrongly understood as failure in data bits. In this way, the multilevel architecture becomes the solution for this issue. The classical solutions for mitigating the effects of SEE are divided in two main categories: reconfiguration-based (i.e. EDAC) and redundancy-based solutions (i.e. TMR).

- The first solutions restore the original values in memory as soon as possible after a SEE event by means of error detection and correction codes, [7], [28], [29], [30], [33].

- The second group masks the SEE effects, isolating the circuit outputs from internal SEUs by means of a voting process, which means that the effect of an error is not transmitted to the output signals. This scheme requires extra cost in terms of components, weight, area and power dissipation.

This work is based on a hybrid system that reduces the overheads associated with the classical fault tolerant solution, TMR plus scrubbing. The proposal mainly couples a (SEC), a reconfiguration-based technique, to a Double Modular Redundancy (DMR) method (as an improved application of the TMR approach), as explained later. SEC functions are stored in the static memory, which is vulnerable to radiation. In order to achieve fault tolerant capability, the SEC function is embedded within an HSB Multilevel Architecture, a new concept based on HSB. HSB is simply a few bits that contain essential information from which errors in original data can be retrieved. They must be, somehow, physically modified to prevent vulnerability to radiation.

Multilevel architecture allows concatenating two or more levels of checksum in which every level only protects the redundant checksum bits of previous levels. That means, data bits of bitstreams are protected by the checksum bits obtained during codification while the own checksum bits will be protected by other level of codification. In case of a failure within data bits, checksum bits will be used to correct the failure while if a failure occurs within checksum bits, the multilevel architecture will be able to manage the failure and correct it without any modification in data bits.

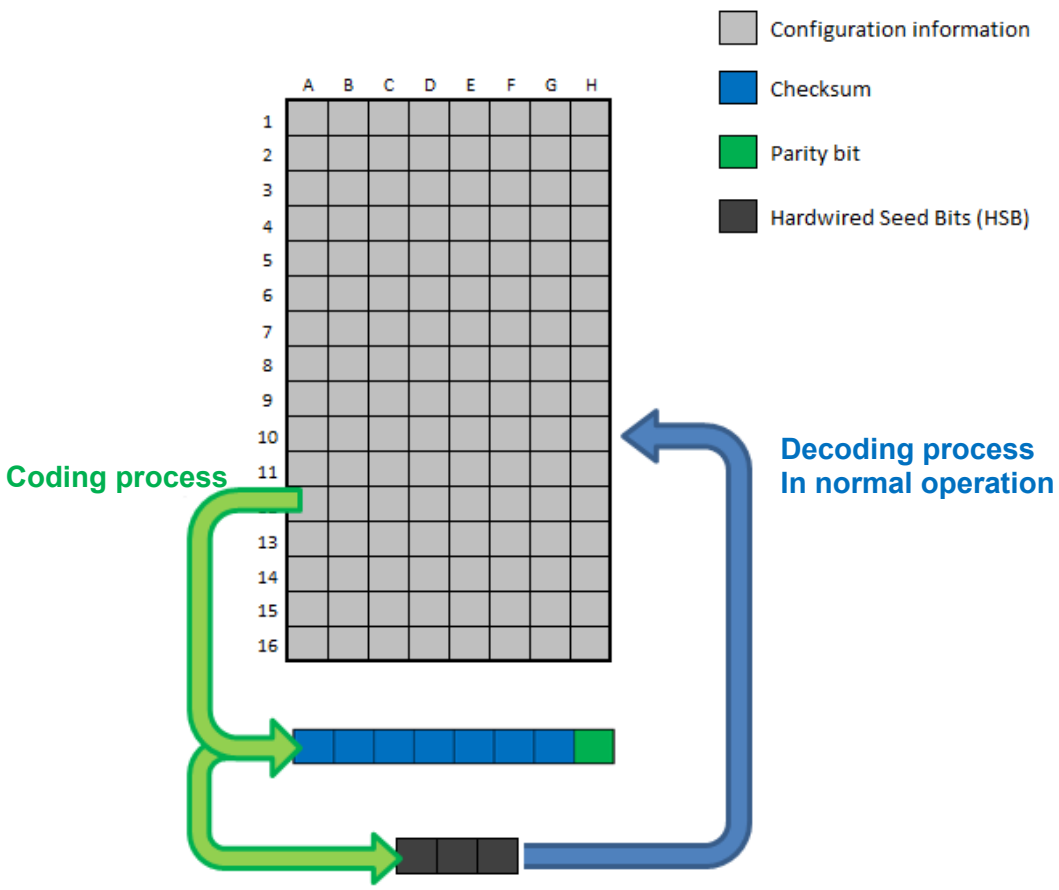


Figure 4.19. Multilevel structure. Seed is a permanent reference

Taking a field-programmable gate array (FPGA) as an example, its static memory originally stores the configuration data for full FPGA functionality. The proposed architecture, Fig. 4.19, adds in the Static Memory:

- The scrubbing function: Logic data that includes the scrubbing function itself and the additional set of bits which are required for the SEC function to work properly (for instance, the parity bits for error detection and reference bits for correction).
- The redundant data: Simply duplicates the scrubbing function.

On the other hand, from the operational point of view, there are 3 levels to be considered:

- Data Level: Composed of the original FPGA configuration static data and the duplicated SEC functions. This SEC separately implements detection (parity) and correction (a simple systematic linear block codification, [6], [7]). Correction is activated only after an error is identified in order to optimize de operation.

- Checkword Level: Comprises the additional set of bits which are required for the SEC function to work properly (parity and reference bits)

- HSB Level: The Checkword level is the reference for the SEC function to work properly to retrieve the Data Level. If an upset occurred in the Checkword level, the SEC function would not be able to retrieve it. This is why the HSB is added as the reference bits for the SEC function to retrieve the Checkword level. In other words, the HSB is the Checkword level of the Checkword level.

The point here is that, due to the exponential ratio between the amount of data to be retrieved and the size of the required SEC reference, the number of bits of the HSB turns out to be extremely low after compressing twice the main data.

There is still a risk of having an upset in the HSB, although this risk has been dramatically reduced in accordance with the exponential ratio. In order to make the risk vanish, the HSB can be hardened, for instance, by physically linking it to a value.

The capital function of multilevel is the protection of the checksum bits required to keep the capability to correct a single error bit anywhere in data bitstream. Top level checksum bits are called “seed bits” because they include enough information to protect and to restore any single error in code of lower levels. As an example of procedure, two steps are performed graphically:

1. Having the following Data bits as an example, it is contained in a word of 31 bits: “1011111001111111100010010101000”.

2. First computation obtains a checksum of 5 bits “00111”, being the complete word “**1011111001111111100010010101000<u>00111</u>**”. As shown in Figure 4.20, complete word including the checksum bits is composed by a total of 36 bits. These 5 checksum bits are able to protect the original data bits from any single error within those 31 bits, not in the last 5 redundant bits.

   Next Figure 4.20 shows how to construct each checksum bits to obtain “00111” by means checksum computation:

   - C1 covers all bit positions which have the least significant bit set to 1: bit D1, D3, D5, D7, D9, etc.

- C2 covers all bit positions which have the second least significant bit set to 1: bit D2, D3, D6, D7, D10, D11, etc.

- C3 covers all bit positions which have the third least significant bit set to 1: bits D4–D7, D12–D15, D20–D23, etc.

- C4 covers all bit positions which have the fourth least significant bit set to 1: bits D8–D15, D24–D31, D40–D47, etc.

- C5 covers all bit positions which have the fifth least significant bit set to 1: bits D16-D31, D48-D63, etc.

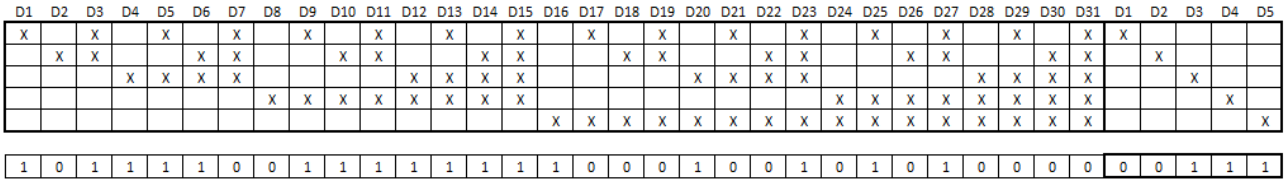


Figure 4.20. Multilevel structure. Lowest level, data and checksum bits

3. Any error injected in those following bits, Checksum bits, could cause a wrongly correction of any data bit as commented previously. For this reason, the capital issue in this architecture is to protect by other means, the bits of checksum. In this case, other checksum codification is included in a second level. Therefore, the result obtained in the first step of calculation, “00111”, is used as data input for second level for checksum computation.

4. Then, the second level of computation is performed only for “00111” as data input. The computation gives a new checksum data that is “010”, being the complete new checksum word of this second level: “**00111<u>010</u>**”. This computation is performed in same way as explained in Figure 4.20 with a reduced number of checksum bits. Those new bits obtained in this second level are called “Seed bits”.

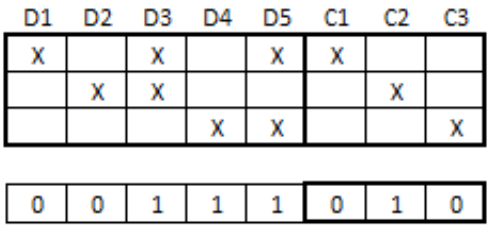


Figure 4.21. Multilevel structure. Seed level, checksum and seed bits

5. Once both level computations are performed, the complete word obtained is defined as follows, data + checksum1 + checksum2: "**1011110011111111000100101010000<u>00111010</u>**". The complete word is then composed by 31 Data bits + 5 Checksum bits + 3 Seed bits = 39 bits. The complete frame of data with three levels is shown in Figure 4.22.

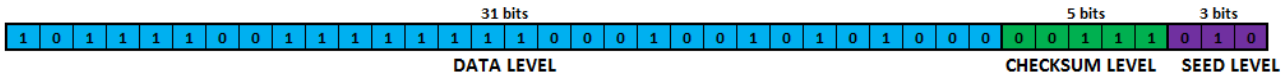


Figure 4.22. Multilevel structure presented in a single frame

6. Those last three bits are saved as Seed bits are used to protect the capability to correct errors anywhere in data bits and also the checksum bits of first level.

In this multilevel architecture, these Seed level bits obtained after two checksum computations are permanently saved by a physical implementation in PCB, called Hardwired Seed Bits (HSB). They keep the capability of error correction as shown in Figure 4.22.

Innovation of the proposed solution is a system based on a fault-tolerant pyramidal reference, Hardwired Seed Bits (HSB), based on multilevel checksum that is an iterative and periodic functionality performed in background of main code. Scrubbing is executed periodically depending on criticism of system functionality and SEU rate predictions of the intended mission, as will be discussed later on.

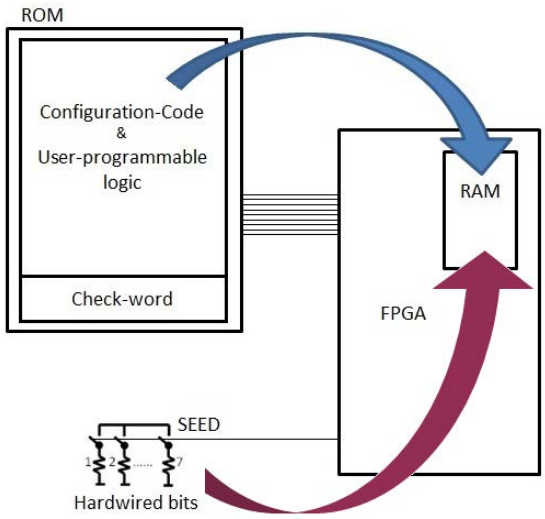


Figure 4.23. Multilevel structure physical implementation.
Both checksum levels and Hardwired Seed Bits (HSB)

Main aim of multi-level system is to protect the checksum bits of previous levels. The seed level bits are permanently saved by a physical implementation of these bits called Hardwired Seed Bits (HSB) keeping the error correction capability of multilevel checksum. This fault-tolerant permanent reference, HSB, includes enough seed information able to restore a single error of code in each scrubbing cycle.

HSB are obtained in Development process by means of multilevel checksum computation while Normal operation uses Decoding Scrubbing function for detecting & correcting errors, shown in Fig 4.19. The pyramidal architecture is intended to perform just decoding scrubbing processes during normal operation due to HSB is a non-modifiable information by any external agent once equipment is delivered to the customer. Codification processes for different levels would be performed by the supplier of equipment in its facilities where HSB are physically implemented according to bitmap saved in memory.

This permanent reference, HSB, includes enough seed information able to restore any potential SEU error in code. The multilevel architecture is intended to perform sequentially decoding scrubbing processes during normal operation for detecting & correcting errors. This encoding process allows to reduce the final number of checksum bits in each subsequence level. In this way, as Table 4.4 shows, up to 32.767 data bits could be protected by 15 Checksum bits that will also be protected by only 4 Seed bits. The range of this multilevel architecture is scalable by increasing the range of Seed bits used. Same scrubbing function logic could be implemented to detect & correct failures for computing both levels.

Seed bits	m	2	3	4	5
Checksum level bits	(2^m)-1	3	7	15	31
Data level bits	2^((2^m)-1)-1	7	127	32.767	2.147.483.647

Table 4.4. Maximum capability for multilevel HSBs

4.5 TIME OPTIMIZATION BASED ON PARITY

As described in previous paragraphs, multilevel architecture is composed by three levels: Data level, checksum level and Seed level. Taking the opportunity of direct protection from the permanent reference of HSB, checksum level includes now a parity bit only of the data bits. In this case, any single failure, SEU/MCU, could be detected by just a parity computation of data-bit level. As commented in Chapter 2. Parity is a low latency process able to detect an odd number of bit errors. Therefore, a parity bit has been added to the end of the bitstream of checksum-word that indicates whether the number of bits in the data-input with the value one is even.

The parity bit ensures whatever the bitstream, a single or any other odd number (i.e., three, five, etc.) of errors within the data bitstream. An even number of flipped bits will make the parity bit appear correct even though the data is erroneous being no able to detect the error. Therefore, this parity bit added to checksum level could be used to detect SEU within data but this architecture based on Parity bit detection will fail in case of an MCU event in which any number of bits, up to 12 bits, could be upset simultaneously.

A numerical example is based on a Data bit set based on 16 Kbits of an SRAM memory). This size of memory could be protected according to Eq. 4.1 to 4.10 and also the data included in Table 4.4. Table 4.5 includes the calculation required to protect this memory size as follows:

- Data level contains 16.383 bits.
- Intermediate Checksum level computes a total of 15 bits, divided into 14 checksum bits available to protect the data level and 1 parity bit computed from data level bits.
- Seed level implemented physically requires only 4 HSB.
- Total redundant bits added to protect the data bits is approximately a 0,12%, that is (14+1+4)/16.383.

In this way, an update of Table 4.4 is performed because of the introduction of the Parity bit in the intermediate checksum level as shown in Table 4.5. Architecture ranged to 4 Seed bits could protect an intermediate Checksum level composed by 15 bits, being 1 bit defined for parity bit and 14 bits defined to protect via checksum computation the data level.

Last row of Table 4.5 shows the percentage of redundant bits required in the multilevel architecture. It is important to highlight that an architecture whose range of HSB is 3, m = 3, is not effective because of low number of bits available in data level. For any other multilevel architecture with a range of HSBs higher than 5, m > 5, the percentage of redundant bits is almost negligible.

Seed bits	m	3	4	5
Intermediate	Parity	1	1	1
Checksum level bits	(2^m)-2	6	14	30
Data level bits	2^((2^m)-2)-1	63	16.383	1.073.741.823
Redundant bits (%)		15,87	0,12	0,00

Table 4.5. Updated maximum capability for multilevel HSBs

The parity bit located within the checksum level is protected directly by HSB. In this way, after checksum level is validated without errors by HSB, the error detection is easily obtained by checking the parity bit that is a low-weight detection procedure. This novel approach gives the capability to separate the procedure for error detection from the procedure for error correction that takes much longer computation timing.

Every iteration cycle, a new frame is masked and restored from data input in memory matrix and computed by means the scrubbing function. This novel approach gives the capability to separate the procedure for error correcting (A) from the procedure for error detecting (B) that takes much lower computation timing. Equation 4.11 that rules the timing spent to compute whole memory depends on number of frames corrupted, x, lower than number of frames, ID2:

$$\textbf{Timing} = \mathbf{A \cdot x + B \cdot ID^2} \qquad \text{Eq. 4.11}$$

The Detection time (B in Eq 4.12) in a frame is typically faster than Correction time (A in Eq 4.12). Therefore, this technique only launches the correction procedure in a frame when an error is detected by the parity bit of frame [7]. Thanks to this, the computation time is hugely reduced because in some EDACs (as Hamming or BCH), detection & correction are coupled being correction procedure performed every time.

According to SEU/MCU rate in airborne electronics, Chapter 2, most of times of operation, there is no error found in memory. Therefore, the separation of procedures obtained in this novel approach between Correction (A) and Detection (B) procedures gives a significant timing reduction when no error is detected.

CHAPTER 5: PROPOSED SOLUTION: DESIGN PROCEDURE

MEAN TIME BETWEEN UPSET

RELIABILITY THRESHOLDS

5.1 DESIGN CRITERIA: MEAN TIME BETWEEN UPSET

This multilevel Framed HSB architecture is a Reconfiguration-based solution that tries to restore as soon as possible the original values in memory after SEE event. This method introduces a limited overheads corresponding to the logic required to periodical control the bitstream rewritten procedure. State-of-the-art of different FPGA suppliers allows online rewriting for a short period of time to rewrite the faulty part from a hard copy of the memory.

As a reconfiguration-based solution, the proposed Architecture has a scrubbing function that detects and corrects the possible SEU/MCU induced in each frame. The interval selected between scrub cycles should be referenced to the expected probabilistic upset rate for a given application or mission, and may be fairly infrequent. Therefore, the scrubbing cycle should be designed according to following condition, equation 5.1:

$$\text{Scrubbing time} < \text{MTBU} \qquad \text{Eq. 5.1}$$

Where MTBU is defined in this work as Mean Time Between Upset. Equipment operating clocking cycle is typically fixed at some orders of magnitude lower than MTBU. Therefore, scrubbing operation will be typically performed thousands of times below the probabilistic MTBU. If previous condition is compliant in the proposed design, there is no probabilistically way to accumulate errors in memory assuring only one error, SEU, affected per cycle. As a rule of thumb, scrubbing cycle should be fixed to one order of magnitude lower than the probabilistic MTBU.

The scrubbing time is defined according to the radiation of natural space environment at A/C flight altitudes. At these A/C flight altitudes (40.000 fts) below “Pfotzer maximum”, most of the particles are neutrons created from nuclear reactions, n-^{28}Si or n-B^{10}, and they affect to CMOS devices. They deposit charge in silicon matter of ICs. Several experiments have verified the fairly linear relationships between flux, critical charge (cell capacitance) and operational frequency versus error rate. The most important way to analyze how SEU can affect to a system is given by the fairly linear relationship between the flux and error rate as following equation rules:

$$\boldsymbol{SEU\ rate = Flux \cdot \sigma \cdot \mathrm{Nbits}} \qquad \text{Eq. 5.2}$$

Flux is defined as the atmospheric particles rate that hits an electronic area over Energy spectrum (MeV). It is measured as particles/cm^2/s. For aircraft applications, the typical neutron flux value is from 1.5 to 2.5 n/cm^2/s. These data

are commonly agreed between different standards such as JEDEC, IEC and ABD100. Several avionics equipment specifications request a peak up to 3,3 $n/cm^2/s$.

"Plateau cross section" represented as σ, is determined empirically. It is defined as the SEE susceptibility of an electronic device to ionizing radiation. This data should be obtained by an experimental test in each component along a wide range of energy transferred to the device. The cross section highly depends on the internal design of bits. Approximate values for current technology, extracted as an average of Weibull distributions of different components, are given for a rough analysis that follows:

$$\sigma_{NEUTRON} \approx 10^{-13}\ cm^2/bit$$

The third important factor in the SEU rate equation is the total amount of bits involved in the working system. Number of bits saved in the system memory on board and the type of memory used are the key characteristics to determine SEU failure rate. SEU rate is proportional to the number of bits used in system. Thus, according to previous equation, Eq 5.2, a baseline study, 128 Kbit RAM memory, has been selected. Therefore, the SEU rate calculation is:

$$\text{SEU rate} = 3{,}3\ n/cm^2/s \cdot 10^{-13}\ cm^2/bit \cdot 2^{17} = 1{,}5 \cdot 10^{-4}/h \qquad \text{Eq. 5.3}$$

Equation 5.3 defines that the SEU rate is $1{,}5 \cdot 10^{-4}/h$ which is not restrictive in terms of memory scrubbing cycle timing (one failure every 6.422 operating hours) but it reveals that the architecture shall include redundancies for this functionality if it is categorized as Catastrophic or Hazardous in System Safety Assessment of equipment.

System Safety Assessment of equipment analyses the criticality (or severity) for all functionalities of equipment and identifies a set of quantitative and qualitative safety requirements for the critical functionalities of system. These safety requirements can be directly associated with the DAL (Design Assurance Level) assigned to the equipment. In case of failure of a function that is defined as Catastrophic or Hazardous, there is no possible operation that could be performed by crew in order to recover the operation. The criticality (or severity) of any function would be reduced if a crew manual operation could be performed when a failure occurs. The DAL defines the rigor that shall be followed during the development process of any equipment in order to avoid failures that could cause critic events. They are defined through different levels, from Level A through Level E corresponding to the five classes of failure conditions: Catastrophic, Hazardous, Major, Minor and No effect. They have an

associated quantitative safety objective in terms of probability of occurrence per flight hour:

- Catastrophic: Failure Rate **< 10^{-9}** as Flight controls, engine controllers.
- Hazardous: Failure Rate **< 10^{-7}** as Navigation and Radio-Navigation.
- Major: Failure Rate **< 10^{-5}** as Voice communications and Displays.
- Minor: Failure Rate **< 10^{-3}** as Maintenance or Monitoring.
- No effect, as Video Systems and Coffee makers.

For example, the safety range of a catastrophic failure condition means that it shall be extremely remote and it shall not be caused by any single failure. This condition, extremely remote, means that a complete A/C fleet shall accumulate more than 10^9 Flight Hours without a catastrophic failure condition in the whole fleet.

5.2 PROBABILITY / RELIABILITY CALCULATIONS

According to the results obtained from Eq. 5.3, the SEU rate for the baseline example, 128 Kbit memory, is $1,5 \cdot 10^{-4}$/h. Probability calculations are estimated based on the number of Operating Hours (OH) in flight in the whole aircraft fleet. Therefore, a typical aircraft fleet could be approximately 1.000 aircrafts. Airbus A320 family has built 7.700 units (includes A319, A320 and A321), Airbus A340 family built 400 units and the Airbus A330 family built 1300 units

Taking the failure rate value of $1,5 \cdot 10^{-4}$/h for 128 Kbits memory, that is equal to 6.422 Operating Hours (OH) in flight under solar radiation for the complete aircraft fleet. Statistically, every 6,422 OH accumulated, a SEU / MCU impact could be produced at any aircraft of the complete fleet. Therefore, an SEU / MCU impact per aircraft could be defined as 6,422 OH / 1000 aircrafts of complete fleet, that is equal to a probabilistic failure every 6.42 hours per aircraft. Finally the scrubbing cycle could be also defined as one order of magnitude lower than the probabilistic failure rate, MTBU, that is 6,42h / 10 = 0,642 h = 39 min. If this value is also extrapolated to the complete A320 family (more than 7.700 aircrafts), the MTBU would be reduced to 0,091h = 5,5 min = 330 seconds.

Thus, the criticality of scrubbing function is not determined by the failure rate in terms of timing constraint because scrubbing period is always much lower than 39 min. In fact, the equipment operating cycle is fixed at some orders of magnitude lower, for example 10us (100KHz). Therefore, scrubbing operation will be typically performed thousands of times within the minimum probabilistic MTBU.

In this way, the operating timing requested to perform the scrub of the complete memory would be defined by the acceptable time in which the most critical functionality of system can work in a non-intended way, let's say, several operating cycles, although this value depends directly on the nature of system. In the Landing Gear Retraction system, whose max operating timing is defined as 2 to 3 seconds to extend, the system could work in a non-intended way during several cycles although depends on the activities or monitoring that are performed in parallel to the main function of Main Landing Gear. In a Primary Flight-Control system, whose mechanical operating performances is up to 20Hz, it will request much lower failure timing.

Thus, according to SEU rate equation, Eq 5.2, different ranges of memory are used to obtain the probabilistic SEU rate depending of memory sizes for the peak value of neutron flux, 3,3 n/cm^2/s:

- 128 Kbits: SEU rate = 3,3 n/cm^2/s · 10^{-13} cm^2/bit · 2^{17} = **$1{,}5 \cdot 10^{-4}$/h**
- 1 Mbits: SEU rate = 3,3 n/cm^2/s · 10^{-13} cm^2/bit · 2^{20} = **$1{,}2 \cdot 10^{-3}$/h**
- 1Gbit: SEU rate = 3,3 n/cm^2/s · 10^{-13} cm^2/bit · 2^{30} = **1,27/h**

Typical normalized rate value (1 n/cm^2/s) is expressed by SEU rate per 1Gbit per day. In this case, a maximum SEU rate between 1 to 10 bits flip per day in 1GBit [2] is statistically considered at altitudes below Pfotzer maximum.

- 1Gbit: SEU rate = 1 n/cm^2/s · 10^{-13} cm^2/bit · 2^{30} · 24 = **10/day**

For the baseline study, 128 Kbit RAM memory, SEU rate reveals a failure rate of $1{,}5 \cdot 10^{-4}$/h requiring redundancies in system if it is categorized as Catastrophic (DAL A) or Hazardous (DAL B). Thus, according to SEU rate equation, Eq 5.2, a baseline study for a 128 Kbit RAM memory has been performed depending of neutron flux and also the possible type of radiation hardened SRAM. Then, the SEU rate calculations are:

- Typ. min flux: SEU rate = **1,5** n/cm^2/s · 10^{-13} cm^2/bit · 2^{17} = **$0{,}708 \cdot 10^{-4}$/h**
- Typ. max flux: SEU rate = **2,5** n/cm^2/s · 10^{-13} cm^2/bit · 2^{17} = **$1{,}18 \cdot 10^{-4}$/h**
- **Peak flux: SEU rate = 3,3 n/cm^2/s · 10^{-13} cm^2/bit · 2^{17} = $1{,}56 \cdot 10^{-4}$/h**
- Hardened: SEU rate = 3,3 n/cm^2/s · **10^{-15}** cm^2/bit · 2^{17} = **$1{,}56 \cdot 10^{-6}$/h**

Table 5.1 extends this concept to other memory sizes. It shows the probability calculations performed of the SEU rate value obtained taking into account the worst scenario in terms of neutron flux, that is 3,3 n/cm^2/s and the standard cross-section of memory devices is used based on $1 \cdot 10^{-13}$ cm^2/bit in order to obtain the worst probabilities for SEU/MCU.

	SEU rate calculation			
	Memory size	Rate at 1.5 n/cm2/s [1/h]	Rate at 2.5 n/cm2/s [1/h]	Rate at 3.3 n/cm2/s [1/h]
	1Kb	5.53E-07	9.22E-07	1.22E-06
	2Kb	1.11E-06	1.84E-06	2.43E-06
	4Kb	2.21E-06	3.69E-06	4.87E-06
	8Kb	4.42E-06	7.37E-06	9.73E-06
	16Kb	8.85E-06	1.47E-05	1.95E-05
	32Kb	1.77E-05	2.95E-05	3.89E-05
	64Kb	3.54E-05	5.90E-05	7.79E-05
REF	128Kb	7.08E-05	1.18E-04	1.56E-04
	256Kb	1.42E-04	2.36E-04	3.11E-04
	512Kb	2.83E-04	4.72E-04	6.23E-04
	1Mb	5.66E-04	9.44E-04	1.25E-03
	2Mb	1.13E-03	1.89E-03	2.49E-03
	4Mb	2.26E-03	3.77E-03	4.98E-03
	8Mb	4.53E-03	7.55E-03	9.97E-03
	16Mb	9.06E-03	1.51E-02	1.99E-02
	32Mb	1.81E-02	3.02E-02	3.99E-02
	64Mb	3.62E-02	6.04E-02	7.97E-02
	128Mb	7.25E-02	1.21E-01	1.59E-01
	256Mb	1.45E-01	2.42E-01	3.19E-01
	512Mb	2.90E-01	4.83E-01	6.38E-01
	1Gb	5.80E-01	9.66E-01	1.28E+00

Table 5.1. SEU rate calculations for different memory sizes for a standard cross-section 10^{-13} cm^2/bit

These calculations represent the probability to have a SEU/MCU impact according to flux and number of bits of memory. Furthermore, as commented in Chapter 2, according to [13], [14], [35], [36] and [37], the SEU/MCU impacts could induce different MCU patterns (Row x Column). They are characterized by the number of bits affected in spatial distribution in rows and columns in memory. Most events are identified as 1x1, SEU, but the total number of events in MCU are relative large. It affects several rows and columns depending on neutron incidence angle. Different patterns are found as 1x2, 2x1, 2x2, 2x4, 3x2 and 4x1. MCU exhibits a strong dependence on device orientation because increases the total number of bits in patterns shown in Figure 5.1. Patter #1 of Figure 5.1 represents the SEU event while the rest of patterns are the most

typical MCU patterns. Pattern #9 is the one with higher number of bits affected with 8 bits affected in only one event having a maximum range of 4 bits.

Therefore, the theoretical maximum range estimated in a conservative way (maximum number of bits affected in line) is 5 bits while maximum number of bits affected estimated (sum of all bits affected between rows and columns) in MCU is up to 12 bits, [10], [12], [13]. As a conclusion of these two constraints, maximum number of bits affected and maximum span, the worst case pattern would be the patterns #13, as shown in Figure 5.2.

Pattern #	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
MCU span	1	2	2	2	2	2	2	3
MCU number of bits	1	2	2	2	3	3	3	3
Probability (%)	70,34	21,76			5,45			

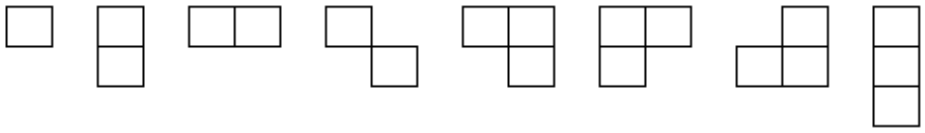


Figure 5.1. MCU patterns and span range

Pattern #	**9**	**10**	**11**	**12**	**13**
MCU span	2	3	4	4	5
MCU number of bits	4	6	5	7-8	9-12
Probability (%)	1,50	0,24	0,48	0,15	0,08

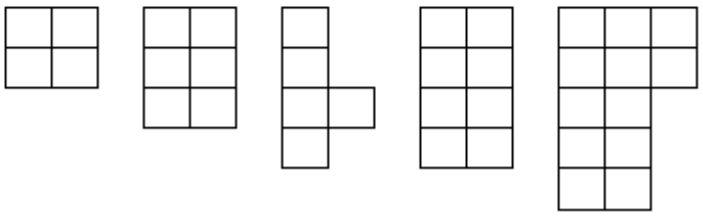


Figure 5.2. MCU patterns and span range

Table 5.2 shows the probabilities for the different number of bits affected by MCU events is obtained from [10] for normal incidence of neutron flux. Therefore, the neutron impact could develop into either a SEU or a MCU and depends on the incidence angle. The ratio works in a range from 70%-30% SEU-MCU ratio for a frontal incidence angle as shown in Table 5.2 to 20%-80% for nearly tangential incidence [22].

MBU range	Prob.
1	70,34%
2	21,76%
3	5,45%
4	1,50%
5	0,48%
6	0,24%
7-8	0,15%
9-12	0,08%

Table 5.2. Probabilities for number of bits affected in MCU event for normal neutron flux incidence.

Therefore, if the distance of HSB Framed architecture, ID, is higher than 5 bits (maximum MCU span), there is no probabilistic case in which the MCU span is higher than the correction capability and therefore, the operational algorithm could guarantee the correction of MCU errors. Table 5.3 represents the probabilities to correct MCU error patterns, Table 5.2, depending of the ID selected in HSB Framed architecture.

	Probability (%)	
	Correctable errors	Non-correctable errors
	ID > span	ID < span
ID = 2	0,9905	0,0095
ID = 4	0,9992	0,0008
ID = 8	1.0000	0
ID = 16	1.0000	0

Table 5.3. MCU range probabilities for different framed architecture distance

Therefore, according to the ID selected in the Framed HSB Architecture, the patterns of Figure 5.1 and 5.2 could lead into correctable / uncorrectable cases depending on the span of the single radiation events. Several terms are defined for probability and rate calculations:

- Impact Rate, **IR,** (1/h): is the number of radiation impacts per operational hour. These impacts could lead to any upset type, either SEU or MCU. This rate can be obtained from Eq. 5.2.

- Impact Probability, **IP**: is the probability of having an Impact within an scrubbing cycle. It can be calculated taking into account the Exposition Time, ExT, define in hours unit, Eq. 5.4. This ExT means the whole time involved to complete a scrubbing cycle (the required time to scrub the whole memory, or the time to complete all the frame iterations). The IP can be obtained from Eq. 5.5:

$$\mathbf{ExT} = \frac{\mathbf{ID}^2}{\mathbf{Iteration\_Freq}} \cdot \frac{\mathbf{1}}{\mathbf{3600}} \ \mathbf{[h]} \qquad \text{Eq. 5.4}$$

$$\mathbf{IP = ExT \cdot IR} \qquad \text{Eq. 5.5}$$

- Single Correctable Impacts Rate, **SC-IR,** (1/h): is the number of radiation impacts per operational hour that are correctable because the ID is enough to cover the MCU span of single radiation event. It is based on the patterns, Fig.5.1 and 5.2, whose span is no greater than the ID, as per Table 5.3. This rate can be obtained from Eq. 5.6:

$$\mathbf{SC\text{-}IR = IR * Prob(\ ID \geq span\ )} \qquad \text{Eq. 5.6}$$

- Single Correctable Impacts Probability, **SC-IP**: is the probability of having a Single Correctable Impact in an Exposition Time. This probability can be obtained from Eq. 5.7:

$$\mathbf{SC\text{-}IP = ExT * SC\text{-}IR} \qquad \text{Eq. 5.7}$$

- Single Uncorrectable Impacts Rate, **SU-IR,** (1/h): is the number of radiation impacts per operational hour that are uncorrectable because the ID is not enough to cover the MCU span. It is based on the patterns, Fig.5.1 and 5.2, whose span is greater than the ID, as per Table 5.3. This rate can be obtained from Eq. 5.8:

$$\mathbf{SU\text{-}IR = IR * Prob(\ ID < span\ )} \qquad \text{Eq. 5.8}$$

- Single Uncorrectable Impacts Probability, **SU-IP**: is the probability of having a Single Uncorrectable Impact in an Exposition Time. This probability can be obtained from Eq. 5.9:

$$\mathbf{SU\text{-}IP = ExT * SU\text{-}IR} \qquad \text{Eq. 5.9}$$

In this study of probability, it is also included the possibility of having consecutive radiation events, which are defined as the event in which there are more than one radiation impacts in memory at same cycle time and affecting to same frame. Therefore, according to the Framed HSB Architecture, these events lead in uncorrectable cases:

- n-Multiple Uncorrectable Impacts Probability, **n-MU-IP**: is the probability of having n-multiple radiation impacts in the same the cycle and affect to same frame. This probability can be obtained from Eq. 5.10. They are considered as uncorrectable because they could lead to having a more than one bit error in the same frame.

$$\textbf{n-MU-IP} = IP^{n} \qquad \text{Eq. 5.10}$$

- n-Multiple Uncorrectable Impacts Rate, **n-MU-IR** (1/h): is the number of n-multiple radiation impacts in the same cycle and affect to same frame, n-MU-IR is expressed in operational hour. It can be obtained from Eq.5.11.

$$\textbf{n-MU-IR} = \frac{n-MU-IP}{Ex.T.} \qquad \text{Eq. 5.11}$$

According to Table 5,3, Eq. 5.6 and 5.8, the main contributor to the calculation of the failure rate is the uncorrectable single impact, SU-IR. This single impact is uncorrectable if the selection of ID in Framed HSB Architecture is not enough to cover the maximum span of all patterns of single impact shown in Fig 5.1 and 5.2. Therefore, a ID non-lower than 8 bits matches with the current maximum span, 5 bits, for a single radiation impact based on current technology [10], [12], [13]. In this way, this Architecture introduces enough capability to detect and correct all single radiation impact, as per Table 5.3.

The setting of Framed Architecture with a ID non-lower than 8 bits improves the failure rate. In this scenario, the main contributor for the calculation of the failure rate is the n-Multiple Uncorrectable Impacts, n-MU-IR, where some impacts occur in same frame at same iteration cycle. Typically, the contribution of double and triple impact rates, 2-MU-IR and 3-MU-IR, would be the most important ones for the failure rate calculation. For that reason, Figure 5.4 to 5.7 also includes the probability calculations for 2-MU-IP and 3-MU-IP, needed to obtain their failure rates, 2-MU-IR and 3-MU-IR.

This allows to focus the n-Multiple Impact Rates, n-MU-IR, as the main contributors to failure rate of Architecture. Adding to this, an statistical analysis

has been also performed in order to obtain the **Total Failure Probability, TFP,** as per Eq. 5.13. It shall take into account all possible scenarios for n-Multiple Uncorrectable Impacts based on the sum of all possible n-multiple cases, from double consecutive impact to n-consecutive impacts. This probability is calculated in Eq. 5.13 through Eq. 5.16.

$$\mathbf{TFP} = \mathbf{IP^2} + \mathbf{IP^3} + \mathbf{IP^4} + \ldots + \mathbf{IP^n} \quad \text{Eq. 5.13}$$

$$\mathbf{TFP} \cdot \mathbf{IP} = \mathbf{IP^3} + \mathbf{IP^4} + \mathbf{IP^5} + \ldots \quad \text{Eq. 5.14}$$

$$\mathbf{TFP} - \mathbf{TFP} \cdot \mathbf{IP} = \mathbf{TFP} \cdot (\mathbf{1} - \mathbf{IP}) = \mathbf{IP^2} \quad \text{Eq. 5.15}$$

$$\mathbf{TFP} = \frac{\mathbf{IP^2}}{\mathbf{1-IP}} = \frac{\mathbf{2-MU-IP}}{\mathbf{1-IP}} \quad \text{Eq. 5.16}$$

According to Eq. 5.16, the Total Failure Probability, **TFP**, depends on double consecutive impact, 2-MU-IP, and the Impact Probability, IP. Therefore, the **Total Failure Rate, TFR,** of all the possible n-Multiple Uncorrectable Impacts in the Framed HSB Architecture is defined as per Eq. 5.17:

$$\mathbf{TFR} = \frac{\mathbf{TFP}}{\mathbf{Ex.T.}} \quad \text{Eq. 5.17}$$

According to these calculations, Eq. 5.2 to Eq. 5.17, a summary of Framed HSB architecture is performed in Tables 5.4 to 5.7 showing the calculations for the rates and the probabilities for different memory sizes with different ID. The calculations were performed under a worst case scenario consisting in:

- A value of 10 KHz for the clock frequency. Typical clock frequencies for digital airborne systems are orders of magnitude above, >100 KHz.

- Just one frame iteration is performed per clock cycle. Therefore, the Exposition Time can be defined as Eq. 5.18.

$$\mathbf{Ex.T.} = \frac{\mathbf{ID^2}}{\mathbf{clock\_Freq}} \cdot \frac{\mathbf{1}}{\mathbf{3600}} \ \mathbf{[h]} \quad \text{Eq. 5.18}$$

- Worst case flux value was considered (3.3 n/cm^2/s).

- The Architecture may isolate the consecutive radiation events into a set of single radiation event even in the same cycle, if they affect to different frames. Only an issue is created when consecutive radiation events in the same scrubbing cycle affect bits in common frames. This fact improves the Architecture failure probability, though it was not taken into account in this study.

Table 5.4 represents the Framed Architecture with an ID equal to 2. As a reference example along chapter, a memory of 16 Kbits has been selected for comparison between different scenarios. The Impact Rate (IR) of this memory is $1{,}95 \cdot 10^{-05}$ 1/hr taking into account the standard cross-section. The failure rate is improved in two orders of magnitude, $1{,}85 \cdot 10^{-07}$ 1/hr, when Framed HSB architecture is introduced in system due to the capability to detect and correct certain errors (Table 5.3).

Failure Rate calculations

	Memory size	IR 1/h	Correctable	Uncorrectable			
			Single impact SC-IR 1/h	Single impact SU-IR 1/h	Double impact 2-MU-IR 1/h	Triple impact 3-MU-IR 1/h	TFR 1/h
REF	16Kb	1,95E-05	1,93E-05	1,85E-07	4,21E-17	9,10E-29	4,21E-17
	128Kb	1,56E-04	1,54E-04	1,48E-06	2,69E-15	4,66E-26	2,69E-15
	1Mb	1,25E-03	1,23E-03	1,18E-05	1,72E-13	2,39E-23	1,72E-13
	8Mb	9,97E-03	9,87E-03	9,47E-05	1,10E-11	1,22E-20	1,10E-11
	64Mb	7,97E-02	7,90E-02	7,57E-04	7,06E-10	6,26E-18	7,06E-10

Probability calculations

	Memory size	IP	Single impact SC-IP	Single impact SU-IP	Double impact 2-MU-IP	Triple impact 3-MU-IP	TFP
REF	16Kb	2,16E-12	2,14E-12	2,05E-14	4,68E-24	1,01E-35	4,68E-24
	128Kb	1,73E-11	1,71E-11	1,64E-13	2,99E-22	5,18E-33	2,99E-22
	1Mb	1,38E-10	1,37E-10	1,31E-12	1,92E-20	2,65E-30	1,92E-20
	8Mb	1,11E-09	1,10E-09	1,05E-11	1,23E-18	1,36E-27	1,23E-18
	64Mb	8,86E-09	8,77E-09	8,42E-11	7,85E-17	6,95E-25	7,85E-17

ID	2	bits
Frames	4	number of frames
Frequency	10	KHz
Flux	3,3	n/cm2/s
Cross section	1,00E-13	cm2/bit
Exposition time	1,11E-07	1/hr

Table 5.4. SEU rate calculations for Framed architecture, ID = 2

Figures 5.4 to 5.7 also include the probability calculations for 2-MU-IP and 3-MU-IP needed to obtain the failure rates according to the Exposition time, ExT defined for the system working at 10 KHz.

Table 5.5 represents the Framed Architecture with an ID equal to 4. The Impact Rate (IR) of the 16 Kbits memory is $1{,}95 \cdot 10^{-05}$ 1/hrs taking into account the standard cross-section. The main restriction for the calculation of the failure rate is the uncorrectable single impact, SU-IR, because the selection of ID is not enough to correct all single MCUs. Therefore, the failure rate, SU-IR, is then improved to $1{,}56 \cdot 10^{-08}$ 1/hr, that means an improvement of three orders of magnitude less than the same system without the framed architecture.

Failure Rate calculations

	Memory size	IR 1/h	Correctable Single impact SC-IR 1/h	Uncorrectable Single impact SU-IR 1/h	Uncorrectable Double impact 2-MU-IR 1/h	Uncorrectable Triple impact 3-MU-IR 1/h	TFR 1/h
REF	16Kb	1,95E-05	1,94E-05	1,56E-08	1,68E-16	1,46E-27	1,68E-16
	128Kb	1,56E-04	1,56E-04	1,25E-07	1,08E-14	7,46E-25	1,08E-14
	1Mb	1,25E-03	1,24E-03	9,97E-07	6,90E-13	3,82E-22	6,90E-13
	8Mb	9,97E-03	9,96E-03	7,97E-06	4,41E-11	1,96E-19	4,41E-11
	64Mb	7,97E-02	7,97E-02	6,38E-05	2,82E-09	1,00E-16	2,82E-09

Probability calculations

	Memory size	IP	Single impact SC-IP	Single impact SU-IP	Double impact 2-MU-IP	Triple impact 3-MU-IP	TFP
REF	16Kb	8,65E-12	8,64E-12	6,92E-15	7,48E-23	6,47E-34	7,48E-23
	128Kb	6,92E-11	6,92E-11	5,54E-14	4,79E-21	3,31E-31	4,79E-21
	1Mb	5,54E-10	5,53E-10	4,43E-13	3,07E-19	1,70E-28	3,07E-19
	8Mb	4,43E-09	4,43E-09	3,54E-12	1,96E-17	8,69E-26	1,96E-17
	64Mb	3,54E-08	3,54E-08	2,83E-11	1,26E-15	4,45E-23	1,26E-15

ID	4	bits
Frames	16	number of frames
Frequency	10	KHz
Flux	3,3	n/cm2/s
Cross section	1,00E-13	cm2/bit
Exposition time	4,44E-07	1/hr

Table 5.5. SEU rate calculations for frame architecture, ID = 4

Table 5.6 represents the Framed Architecture with an ID equal to 8. In this scenario, the Architecture introduces the capability to detect and correct all single radiation impacts, as per Table 5.3, because the selected ID is higher than maximum span of possible MCU patterns.

Therefore, the main contribution to the failure rate in this scenario, ID = 8, is the consecutive radiation events, 2-MU-IR, where both impacts occur in same frame at same iteration cycle. The failure rate is reduced to $6{,}74 \cdot 10^{-16}$ 1/hrs, that means **an improvement of eleven orders of magnitude** less than the same system without any protection, $1{,}95 \cdot 10^{-05}$ 1/hrs.

The contribution of triple impact rates, 3-MU-IR, to final failure rate is negligible compared to double uncorrectable impact rate, 2-MU-IR. **The calculation of the Total Failure Rate (TFR) for this scenario, that includes all posible cases for n-Multiple Uncorrectable Impacts, Eq. 5.13 to 5.17 reveals that TFR is equal to 2-MU-IR.** Therefore, the unique constraint to the failure rate in this scenario is the double consecutive radiation events, 2-MU-IR.

Failure Rate calculations

	Memory size	IR 1/h	Correctable Single impact SC-IR 1/h	Uncorrectable Single impact SU-IR 1/h	Uncorrectable Double impact 2-MU-IR 1/h	Uncorrectable Triple impact 3-MU-IR 1/h	TFR 1/h
REF	16Kb	1,95E-05	1,95E-05	0,00E+00	6,74E-16	2,33E-26	6,74E-16
	128Kb	1,56E-04	1,56E-04	0,00E+00	4,31E-14	1,19E-23	4,31E-14
	1Mb	1,25E-03	1,25E-03	0,00E+00	2,76E-12	6,11E-21	2,76E-12
	8Mb	9,97E-03	9,97E-03	0,00E+00	1,77E-10	3,13E-18	1,77E-10
	64Mb	7,97E-02	7,97E-02	0,00E+00	1,13E-08	1,60E-15	1,13E-08

Probability calculations

	Memory size	IP	Single impact SC-IP	Single impact SU-IP	Double impact 2-MU-IP	Triple impact 3-MU-IP	TFP
REF	16Kb	3,46E-11	3,46E-11	0,00E+00	1,20E-21	4,14E-32	1,20E-21
	128Kb	2,77E-10	2,77E-10	0,00E+00	7,66E-20	2,12E-29	7,66E-20
	1Mb	2,21E-09	2,21E-09	0,00E+00	4,90E-18	1,09E-26	4,90E-18
	8Mb	1,77E-08	1,77E-08	0,00E+00	3,14E-16	5,56E-24	3,14E-16
	64Mb	1,42E-07	1,42E-07	0,00E+00	2,01E-14	2,85E-21	2,01E-14

ID	8	bits
Frames	64	number of frames
Frequency	10	KHz
Flux	3,3	n/cm2/s
Cross section	1,00E-13	cm2/bit
Exposition time	1,78E-06	1/hr

Table 5.6. SEU rate calculations for frame architecture, ID = 8

Table 5.7 represents the Framed Architecture with an ID equal to 16. This architecture introduces the capability to detect and correct all single radiation impact, as per Table 5.3. The unique contribution to the failure rate in this scenario is the double radiation event, 2-MU-IR, as explained also in previous scenario. The failure rate is reduced to $2{,}69 \cdot 10^{-15}$ 1/hrs, that means **an improvement of ten orders of magnitude** less than the same system without any protection, $1{,}95 \cdot 10^{-05}$ 1/hrs. This solution with the ID equal to 16 does not improve the failure rate, 2-MU-IR, respect to the previous scenario, ID = 8, because the Exposition time is higher due to the higher number of frames.

Failure Rate calculations

	Memory size	IR 1/h	Correctable Single impact SC-IR 1/h	Uncorrectable Single impact SU-IR 1/h	Uncorrectable Double impact 2-MU-IR 1/h	Uncorrectable Triple impact 3-MU-IR 1/h	TFR 1/h
REF	16Kb	1,95E-05	1,95E-05	0,00E+00	2,69E-15	3,73E-25	2,69E-15
	128Kb	1,56E-04	1,56E-04	0,00E+00	1,72E-13	1,91E-22	1,72E-13
	1Mb	1,25E-03	1,25E-03	0,00E+00	1,10E-11	9,78E-20	1,10E-11
	8Mb	9,97E-03	9,97E-03	0,00E+00	7,06E-10	5,00E-17	7,06E-10
	64Mb	7,97E-02	7,97E-02	0,00E+00	4,52E-08	2,56E-14	4,52E-08

Probability calculations

	Memory size	IP	Single impact SC-IP	Single impact SU-IP	Double impact 2-MU-IP	Triple impact 3-MU-IP	TFP
REF	16Kb	1,38E-10	1,38E-10	0,00E+00	1,92E-20	2,65E-30	1,92E-20
	128Kb	1,11E-09	1,11E-09	0,00E+00	1,23E-18	1,36E-27	1,23E-18
	1Mb	8,86E-09	8,86E-09	0,00E+00	7,85E-17	6,95E-25	7,85E-17
	8Mb	7,09E-08	7,09E-08	0,00E+00	5,02E-15	3,56E-22	5,02E-15
	64Mb	5,67E-07	5,67E-07	0,00E+00	3,21E-13	1,82E-19	3,21E-13

ID	16	bits
Frames	256	number of frames
Frequency	10	KHz
Flux	3,3	n/cm2/s
Cross section	1,00E-13	cm2/bit
Exposition time	7,11E-06	1/hr

Table 5.7. SEU rate calculations for frame architecture, ID = 16

According to the study of previous scenarios, Table 5.8 includes a summary of the failure rates capabilities from Safety point of view. The limitations of this technique depend on the ID selected for Framed HSB architecture and also the size of the memory. Once Frame architecture is sized to be able to correct all single radiation impact, ID = 8 or higher, the failure rate is significantly reduced.

		Framed Architecture			
	No ID	ID = 2	ID = 4	ID = 8	ID = 16
1 Mb	1,25E-03	1,23E-03	1,24E-03	2,76E-12	1,10E-11
2 Mb	2,49E-03	2,47E-03	2,49E-03	1,10E-11	4,41E-11
4 Mb	4,98E-03	4,94E-03	4,98E-03	4,41E-11	1,77E-10
8 Mb	9,97E-03	9,87E-03	9,96E-03	1,77E-10	7,06E-10
16 Mb	1,99E-02	1,97E-02	1,99E-02	7,06E-10	2,82E-09
32 Mb	3,99E-02	3,95E-02	3,98E-02	2,82E-09	1,13E-08
64 Mb	7,97E-02	7,90E-02	7,97E-02	1,13E-08	4,52E-08
128 Mb	1,59E-01	1,58E-01	1,59E-01	4,52E-08	1,81E-07
256 Mb	3,19E-01	3,16E-01	3,19E-01	1,81E-07	7,23E-07
512 Mb	6,38E-01	6,32E-01	6,37E-01	7,23E-07	2,89E-06

No safety critical

Hazardous

Catastrophic

Table 5.8. SEU/MCU rate calculations

Furthermore, table 5.8 highlights the configurations that could be used for Safety critical applications depending of memory size. Table 5.8 also marks that the Framed HSB architecture allows implementing functionalities in memory categorized as:

- Catastrophic (DAL A) with a minimum probability of $1 \cdot 10^{-9}$ 1/hr.
- Hazardous (DAL B) with a minimum probability of $1 \cdot 10^{-7}$ 1/hr.

CHAPTER 6:EXPERIMENTAL ASSESSMENT

SIMULATIONS

EXPERIMENTAL RESULTS

Different simulations and implementations have been performed in the Multilevel Framed HSB architecture with different ranges of ID and HSB. This architecture has been implemented, simulated in VHDL for different memory sizes, then Synthetized and Placed & Routed into a Xilinx Spartan FPGA to check viability of system. The results obtained after the synthesis and implementation of system with redundant function logics are included in this section.

6.1 FUNCTIONAL SIMULATIONS

Fault-tolerant HSB architecture development has been briefly presented in this Thesis. Several examples of this architecture, Table 6.5, have been implemented and simulated in VHDL for different memory sizes. They have been synthetized and placed & routed into a Xilinx Spartan FPGA to check the feasibility of the system using the Xilinx ISE software. Results obtained depend quantitatively on the type of device in which the algorithm is implemented, although it enables having an example in terms of time and logic consumption for a qualitative assessment.

Scenario 1, 2 and 3 of the Framed HSB architecture have been reported in this chapter including the detailed results of the simulations. Other scenarios (Scenarios 4 and 5) have been included in Table 6.1 and 6.2 for comparison purpose. Main purpose of these scenarios is to show the trends and the limits of the Frame architecture in terms of performance capabilities and overheads. All the scenarios considered in this section were implemented under the FT approach according to Equation 4.1 to 4.10. Table 6.1 represents three scenarios with different ID selection for Framed HSB architecture in a memory of 128 Kbits:

- Scenario 1: 128Kbits memory, ID is defined as 8. Chapter 6.1.1
- Scenario 2: 128Kbits memory, ID is defined as 4. Chapter 6.1.2
- Scenario 4: 128Kbits memory, ID is defined as 2.

Table 6.2 represents three different memory sizes with same ID for Frame architecture:

- Scenario 5: 1 Mbits memory, ID is defined as 8.
- Scenario 1: 128Kbits memory, ID is defined as 8. Chapter 6.1.1
- Scenario 3: 8Kbits memory, ID is defined as 8. Chapter 6.1.3

Xilinx Spartan FPGA was used to compute these bitstreams with errors. These manually modified bitstreams were used in Xilinx Spartan FPGA as a fault generator to demonstrate the retrieval capabilities of HSB architecture against MCU events. Furthermore, six different static memory data patterns were also used in those tests as a base of the static memory data with errors injected [39]: all-ones (AO), all-zeros (AZ), checkerboard (CB), column-stripe (CS), row-stripe (RS) and random board (RB) patterns.

Xilinx ISE software provides the required digital resources in terms of flip-flops, adders and multiplexers, as well as its associated delays. Different MCU error span scenarios (0, 2 and 8 upsets respectively) were considered for the proposed architecture that covers the MCU patterns described in Chapter 2.

Different conclusions can be obtained after the simulations by optimizing the ranges of HSB and ID as shown in Table 6.1 and 6.2. These scenarios are developed to show the MCU correction capabilities of architecture and also the overheads required for a certain protection.

Variable	Concept	Scenario 1 ID = 8	Scenario 2 ID = 4	Scenario 4 ID = 2
ID	**Interleaving Distance**	**8 bits**	**4 bits**	**2 bits**
FN	Frame number	64 frames	16 frames	4 frames
MCU	ID > MCU span	**Yes**	**No**	**No**
	MCU max correction	8 x 8 bits	4 x 4 bits	2 x 2 bits
MCUspan	MCU span bits	5 bits	5 bits	5 bits
MCUbits	Maximum MCU bits	12 bits	12 bits	12 bits
DS	Data size	130.816 data bits	130.816 data bits	130.816 data bits
FB	Frame bits	2.044 bits	8.176 bits	32.704 bits
CHB	Checksum bits per frame	11 checksum bits	13 checksum bits	15 checksum bits
PB	Parity bits per frame	1 parity bit	1 parity bit	1 parity bit
Overhead CH	Number redundant bits in checksum	768 redundant bits (0,6 %)	224 redundant bits (0,17 %)	64 redundant bits (0,05 %)
HSB	Hardwired Seed Bits per frame	4 HSB per frame	4 HSB per frame	5 HSB per frame
Overhead HSB	Number redundant HSB in architecture	**256 HSB (0,2 %)**	**64 HSB (0,05 %)**	**20 HSB (0,02 %)**

Table 6.1. Multilevel Frame HSB architecture range capabilities for same 128 Kbit memory

As Table 6.1 shows, as ID increases, higher MCU range correction capability. Higher distances increase the number of frames available in the structure making also higher the matrix dimension of MCU patterns that could be detected and corrected. Therefore the correction capabilities are defined as:

ID = 2 => 4 frames => 2x2 MCU matrix, up to 4 bits
ID = 4 => 16 frames => 4x4 MCU matrix, up to 16 bits
ID = 8 => 64 frames => 8x8 MCU matrix, up to 64 bits

As ID increases, higher number of checksum bits. For a same memory size, higher distances increase the number of frames available. As higher number of frames in a memory, the number of bits per frame will be lower but the total number of checksum bits will be increased when multiplied by the total number of frames.

ID = 2 => 15 bits per frame => 64 bits in 4 frames
ID = 4 => 13 bits per frame => 224 bits in 16 frames
ID = 8 => 11 bits per frame => 768 bits in 64 frames.

Variable	Concept	Scenario 5 1 Mbit	Scenario 1 128Kbits	Scenario 3 8Kbits
ID	Interleaving Distance	8 bits	8 bits	8 bits
FN	Frames number	64 frames	64 frames	64 frames
MCU	ID > MCU span	**Yes**	**Yes**	**Yes**
	MCU max correction	8 x 8 bits	8 x 8 bits	8 x 8 bits
FN	Frames number	64 frames	64 frames	64 frames
MCUspan	MCU span bits	5 bits	5 bits	5 bits
MCUbits	Maximum MCU bits	12 bits	12 bits	12 bits
DS	**Data size**	**1.048.576 data bits**	**130.816 data bits**	**7.936 data bits**
FB	Frame bits	16.383 bits	2.044 bits	124 bits
CHB	Checksum bits per frame	14 checksum bits	11 checksum bits	7 checksum bits
PB	Parity bits per frame	1 parity bit	1 parity bit	1 parity bit
Overhead CH	Number redundant bits in checksum	960 redundant bits (0,1 %)	768 redundant bits (0,6 %)	512 redundant bits (6,4 %)
HSB	Hardwired Seed Bits per frame	4 HSB per frame	4 HSB per frame	4 HSB per frame
Overhead HSB	Number redundant HSB in architecture	**256 HSB (0,02 %)**	**256 HSB (0,2 %)**	**256 HSB (3,2 %)**

Table 6.2. Multilevel Frame HSB architecture range capabilities for 256 HSB

Furthermore, as represented in Table 6.2, same number of overhead in the HSB could protect a wide variety of memory sizes. As HSB range increases, higher memory sizes could be protected. By increasing the number of HSBs,

capacity of checksum level also increases. Therefore, maximum number of bits that could be included in data-input level is increased as well:

HSB=3 protects up to 7 checksum bits, means up to 63 bits per frame
HSB=4 protects up to 15 checksum bits, means up to 16 Kbits per frame
HSB=5 protects up to 31 checksum bits, means up to 1 Gbit per frame

As number data bit increases, higher overheads in checksum are required due to higher number of bit per frame. The number of checksum bits increases to protect the data per frame. As table 6.2 shows, same number of HSB are able to protect a wide range of checksum bits. That means that same 256 HSB are able to protect from 512 checksum bits in Scenario 3 to 960 checksum bits in Scenario 5.

6.1.1 **SCENARIO 1**

First simulation corresponds to 4088 words of 32 bits each, 128Kbit memory, managed by the Multilevel Framed HSB architecture with an **ID equal to 8bit**. Figure 6.1 represents a subset of the complete memory. The complete memory is composed by the data memory and also 24 words of 32 bits each added for checksum level that also includes the parity bits. The Framed HSB architecture also implements externally 8 words of 32 bits each for HSB protection.

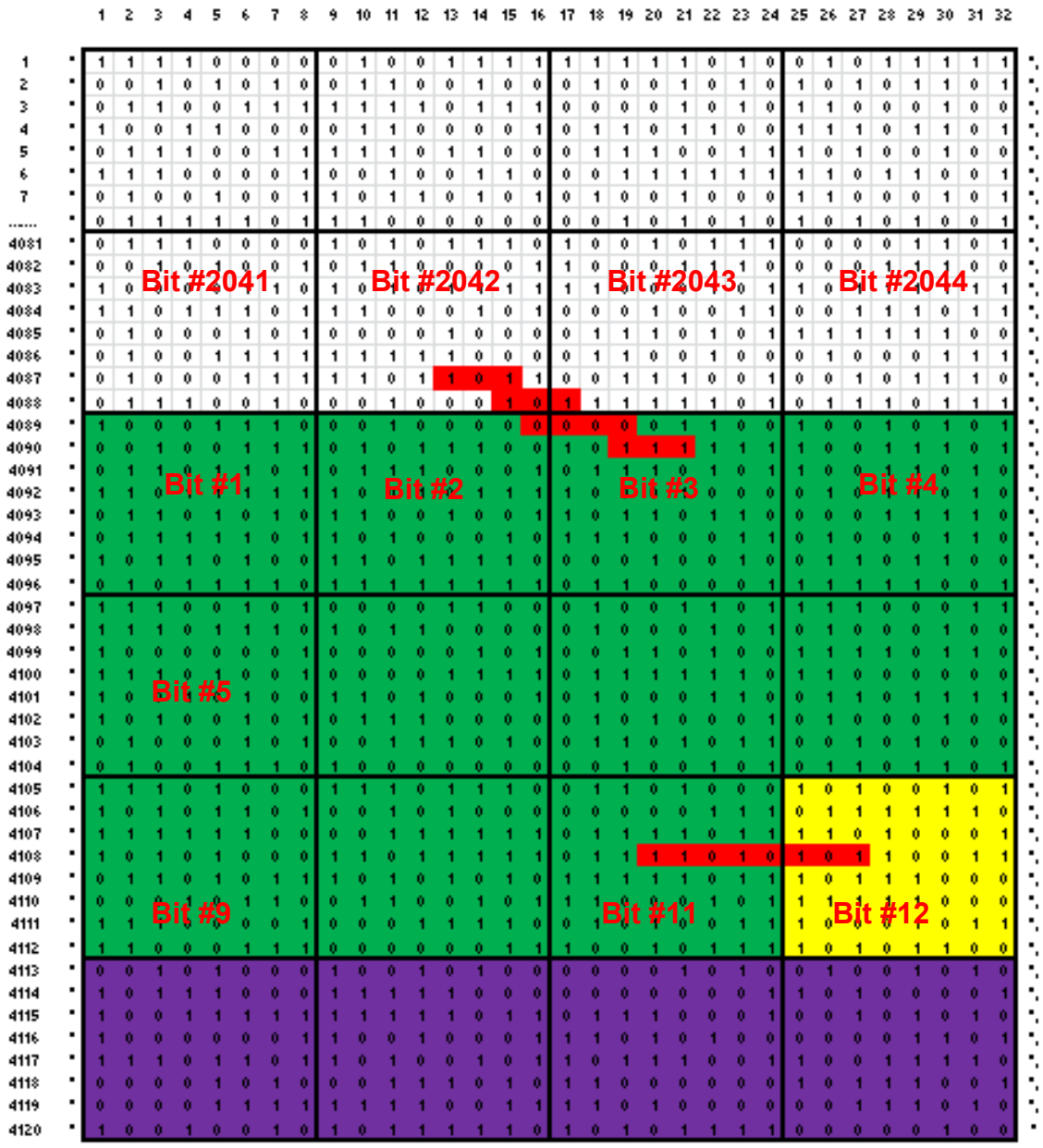


Figure 6.1. 128Kbit memory masked by
Multilevel Framed HSB architecture with distance set to 8

In Figure 6.1, first rows, marked in white color, include the 4088 words of 32 bits for the data, 130.816 bits. The matrix is divided in sections of 8x8 matrix due to the ID selected for this architecture equal to 8 bits in both dimensions. Each bit of this section pertains to a different frame, 64 frames in total. Some sectors of Figure 6.1 are marked with the order for clarification. The sector marked as bit #2041 means that the bit located in the upper part at left of this sector is the bit number 2.041 from frame1 while the bit located in the lower part of right of this sector is the bit number 2.041 from last frame, number 64. According to table 6.1, the scenario 1 could protect up to 2.044 bits per frame.

The architecture values, according to Eq 4.1 to 4.10, could be optimized to match better with the 32 bits (or 64 bit) width memory as shown in Figure 6.1. In a word of 32 bits with a distance, ID, fixed to 8 bits, only 4 of 32 bits word could be used per frame. Therefore, checksum should be divided also in groups of 4 bits per word to match the architecture of 32 bits word and 8 bit distance. Thus, the 11 bit checksum plus 1 bit parity is 12 bits that match with a multiple of 4. That means that each checksum level (parity included) of each frame requires at least 3 words to be allocated in the architecture.

In this way, complete memory is divided in three level as known: Data-input level, Intermediate checksum level and Seed level, they are sized as follows:

- Data level. It is composed by a 4.088x32 bit memory, 128 Kbits memory**.** As shown in Figure 6.1, it covers from row 1 up to row 4.088, marked in white colour.

- Checksum level. The protection of the 130.816 data bits with 64 frames implies 2.044 bits per frame. This requires at least 11 bits to protect the data level. Furthermore, the Frame architecture adds 1 parity bit per frame. Therefore, a total of 12 bits are required for the checksum level per frame. As shown in Figure 6.1, it covers from row 4.089 up to row 4.112. Checksum bits are marked in green color and parity bit is marked in yellow color.

- Seed level. It is composed by 4 bits per frame as shown in Table 6.1. They are required to protect the 12 bits in the checksum level per frame. It covers from row 4.113 up to row 4.120, marked in purple color. This 32 bit words are not part of memory although it is represented together for simplicity.

Once architecture is explained, two MCU events have been also included and marked in red color in Figure 6.1:

- First MCU event is a pure horizontal event that causes an 8 bit upsets in the row 4.108. This is the maximum capability of Multilevel Framed HSB architecture. This MCU event affects the area reserved to checksum and parity. Bits affected are referenced as:
    1. Frame 25, Bit 12, error in Parity
    2. Frame 26, Bit 12, error in Parity
    3. Frame 27, Bit 12, error in Parity
    4. Frame 28, Bit 11, error in Checksum
    5. Frame 29, Bit 11, error in Checksum
    6. Frame 30, Bit 11, error in Checksum
    7. Frame 31, Bit 11, error in Checksum
    8. Frame 32, Bit 11, error in Checksum

- Second MCU event is a skew event that causes a 13 bit MCU. This is within the maximum capability of Multilevel Framed HSB architecture with ID equal to 8 bits. This MCU event affects the area reserved to data and checksum. Bits affected are referenced as:
    1. Frame 1, Bit 3, error in Checksum
    2. Frame 2, Bit 3, error in Checksum
    3. Frame 3, Bit 3, error in Checksum
    4. Frame 8, Bit 2, error in Checksum
    5. Frame 11, Bit 3, error in Checksum
    6. Frame 12, Bit 3, error in Checksum
    7. Frame 13, Bit 3, error in Checksum
    8. Frame 53, Bit 2.042, error in Data
    9. Frame 54, Bit 2.042, error in Data
    10. Frame 55, Bit 2.042, error in Data
    11. Frame 57, Bit 2.043, error in Data
    12. Frame 63, Bit 2.042, error in Data
    13. Frame 64, Bit 2.042, error in Data

A complete simulation of architecture has been performed with the previous MCU event included. The figures 6.2 to 6.7 are included as summary where some variables have been included as monitoring of the working system as:

- **tb_proceso_pyr_dec/clk**: clock of architecture, 50MHz
- **tb_proceso_pyr_dec/status**: status of memory. It is activated if a failure has been detected. It will be kept activated until the full memory (64 frames) is checked.
- **tb_proceso_pyr_dec/error:** it is activated if an error has been detected in frame
- **tb_proceso_pyr_dec/bitnumber:** it points to the number of the bit of the frame that has been corrected.

- **tb_proceso_pyr_dec/bitframe:** it points to the number of the frame where an error has been corrected.

In the simulation shown in Figure 6.2, errors both MCU events are corrected. Figure 6.3 shows the zoom of simulation 6.2. The vertical mark of oscilloscope is set **at 165ns** where an error is detected in frame number 54 being the bit 2.042 of this frame, the one that is corrected in this clock cycle. Other adjacent frames are also detected and corrected as frame 53 and 55. These bits correspond to bits placed in data.

Figure 6.4 shows other zoom of simulation 6.2. The vertical mark of oscilloscope is set **at 405ns** where an error is detected in frame number 2 being the bit 3 of this frame, the one that is corrected. Other adjacent frames are also detected and corrected as frame 1 and 3. These bits correspond to bits placed from checksum level. Furthermore, there are other frames 63 and 64, where an error is detected other bit section that corresponds to bits 2.042, that corresponds to bits placed in data.

Figure 6.5 shows other zoom of simulation 6.2. The vertical mark of oscilloscope is set **at 605ns** where an error is detected in frame number 12 being the bit 3 of this frame, the one that is corrected. Other adjacent frames are also detected and corrected as frame 11 and 13. These bits correspond to bits placed in checksum level.

Figure 6.6 shows other zoom of simulation 6.2. The vertical mark of oscilloscope is set **at 885ns** where an error is detected in frame number 26 being the bit 12 of this frame, the one that is corrected. Other adjacent frames are also detected and corrected as frame 25 and 27. These bits correspond to bits placed in checksum level, in parity section. Furthermore, there are other frames 28, 29, 30, 31 and 32, where an error is detected other bit section that corresponds to bits 11, in the checksum level, no in parity.

Figure 6.7 shows other zoom of simulation 6.2. It is shown how signal status flips to zero because the complete memory, the last 64 frames, has been checked without any error. This signal becomes ‘1’ when an error is detected and keeps in this status up to the complete memory is checked without errors after frame 31 that was the last frame detected with error.

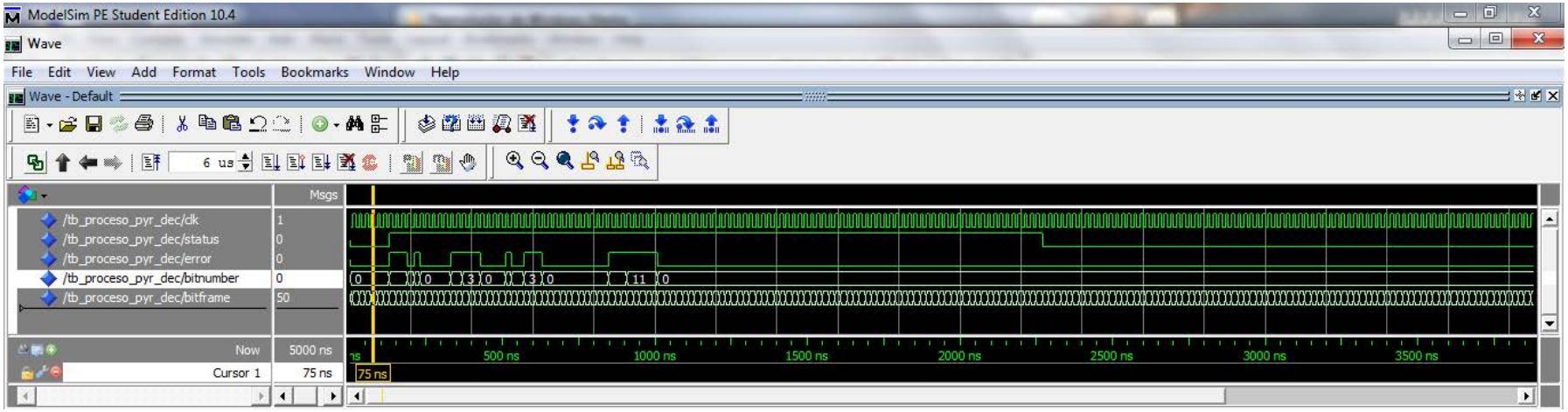


Figure 6.2. Complete simulation of Multilevel Framed HSB architecture with a distance equal to 8

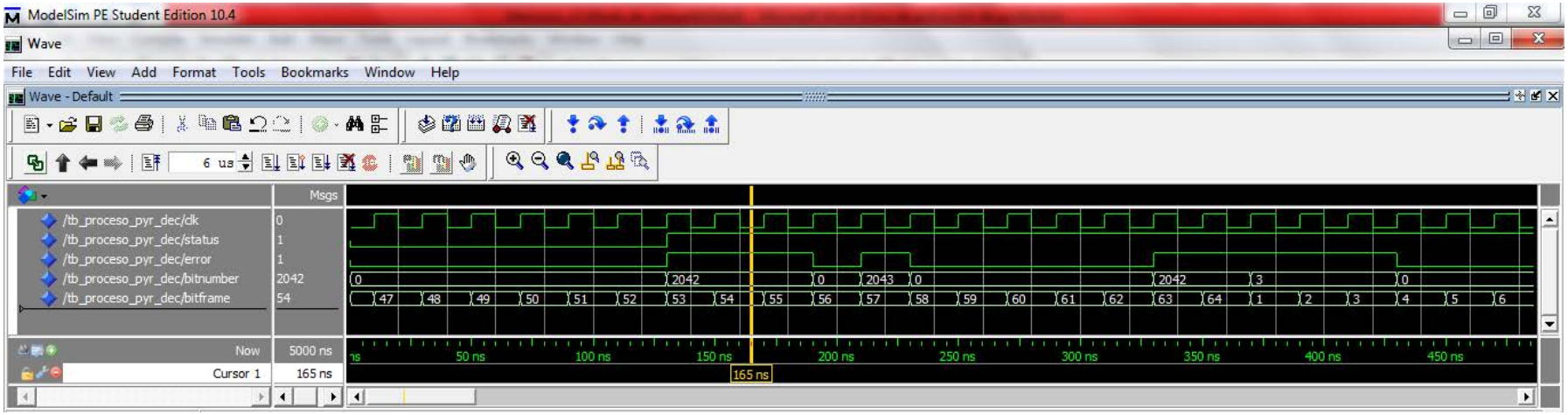


Figure 6.3. First frames detected in failure in data-input level. Mark shows a SEU in Frame 54 in bit 2042 at 165ns.

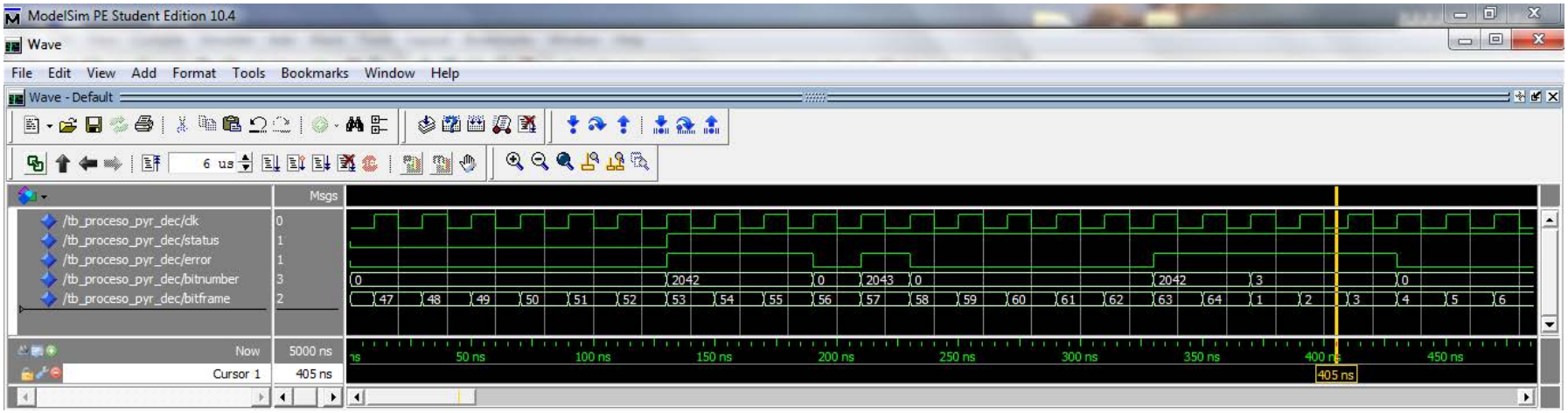


Figure 6.4. First frames detected in failure in intermediate checksum level. Mark shows a SEU in Frame 2 in bit 3 at 405ns.

Figure 6.5. Frames detected in failure in intermediate checksum level. Mark shows a SEU in Frame 12 in bit 3 at 605ns

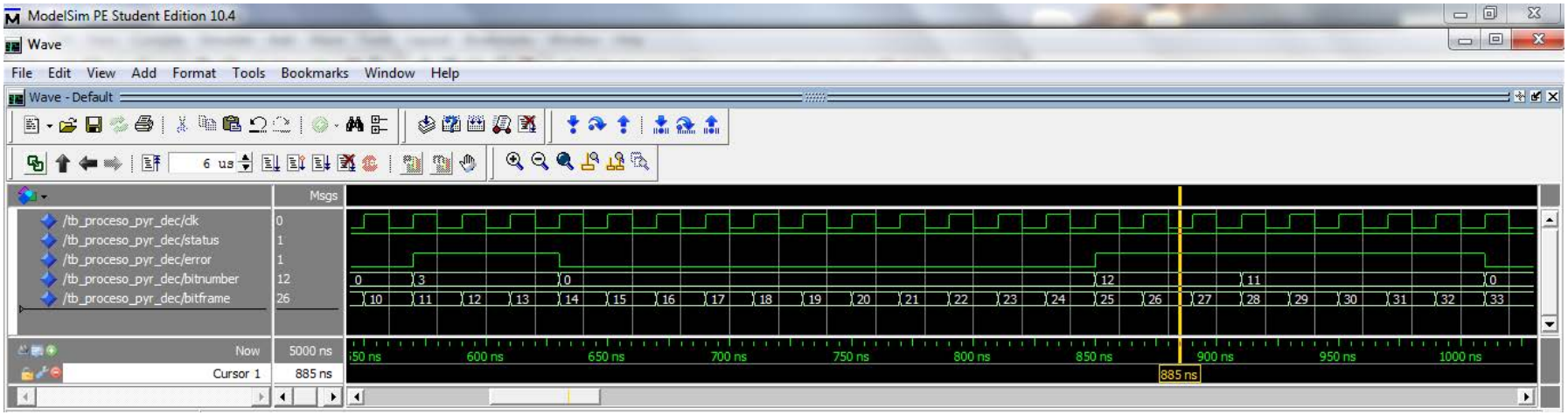


Figure 6.6. Frames detected in failure in Parity. Mark shows a SEU in Frame 26 in bit 12 at 885ns

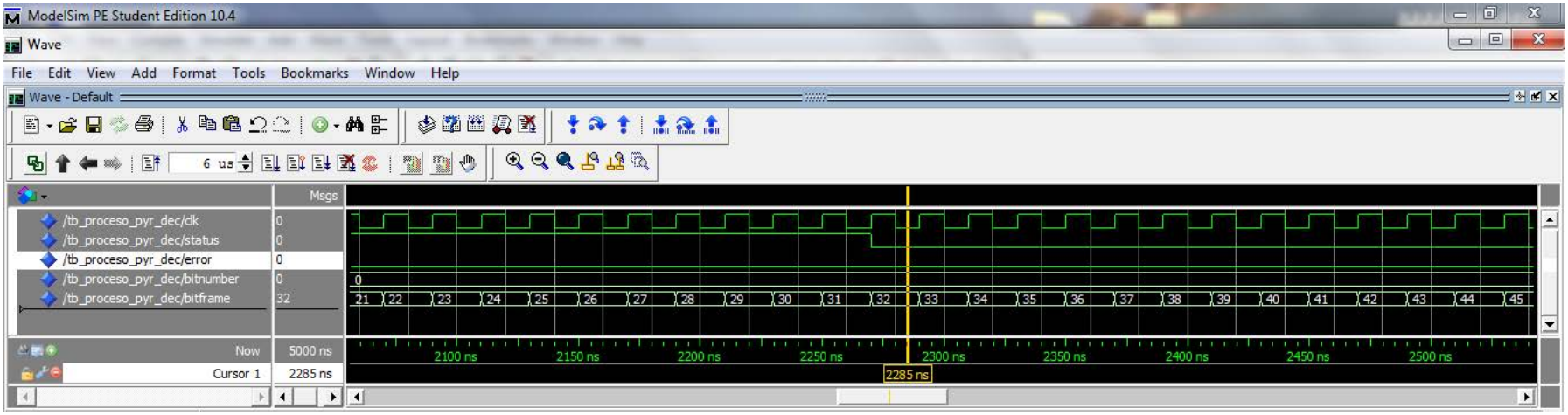


Figure 6.7. Status signal is removed once all frames have been detected without error between frame 31 and 32.

6.1.2 **SCENARIO 2**

Second simulation corresponds to 4088 words of 32 bits each, 128Kbit memory, managed by the Multilevel Framed HSB architecture with an **ID equal to 4bit**. Figure 6.8 represents a subset of the complete memory. The complete memory is composed by the data memory and also 8 words of 32 bits each added for checksum level that also includes the parity bits. The Framed HSB architecture also implements externally 4 words of 5 bits each for HSB protection.

The Framed HSB architecture definition, according to Eq 4.1 to 4.10, is adapted in this scenario to match with the 32 bits (or 64 bit) word width memory, as shown in Figure 6.8. This is non-optimized scenario because there is two sections, bit #14 and #15 (marked in grey) in checksum level that are not used in this configuration and are filled with zeros. This is non-optimized scenario implies to have one more HSB per frame to protect to complete checksum. Theoretically, 13 checksum bits and 1 parity bit could be protected by only 4 HSB as shown in table 6.1 but in order to match with the 32 bits word width memory, a total of 15+1 checksum bits are required. This implies 1 more HSB per frame.

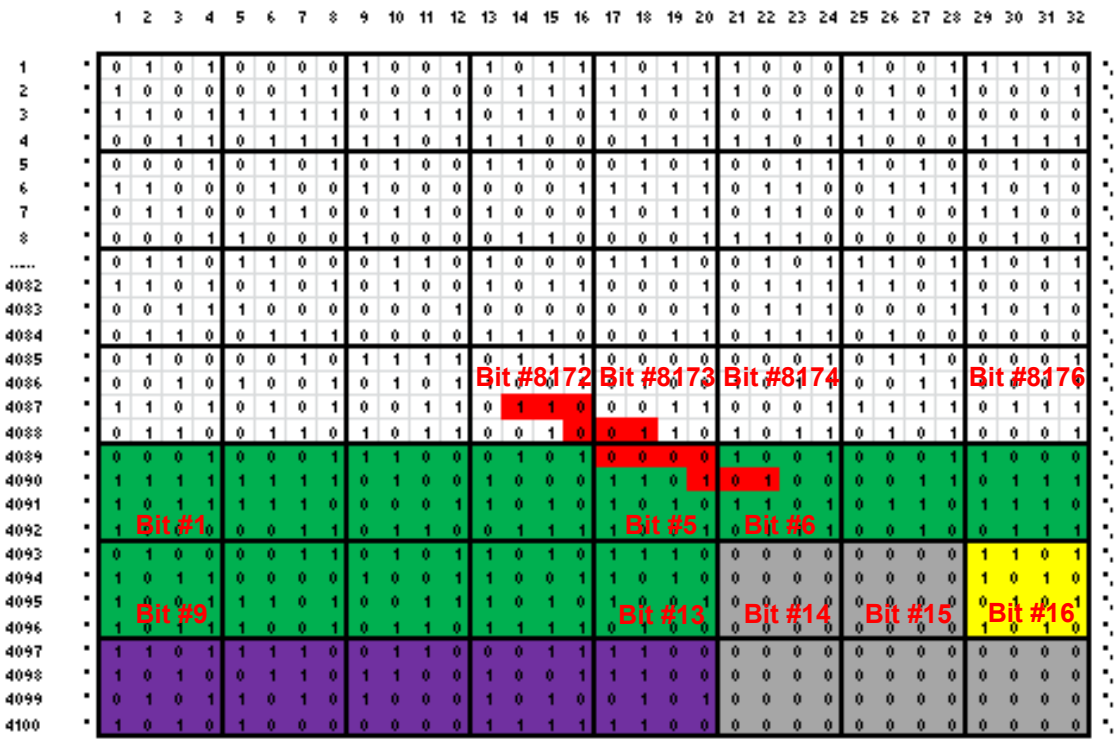


Figure 6.8. 128Kbit memory masked by
Multilevel Framed HSB architecture with distance set to 4

In Figure 6.8, first rows, marked in white color, include the 4088 words of 32 bits for the data-input, 130.816 bits. The matrix is divided in sections of 4x4 matrix due to the ID selected for this architecture equal to 4 bits in both dimensions. Each bit of this section pertains to a different frame, 16 frames in total. In this way, it is **not** assured that all MCU events could be corrected.

In this way, complete memory is divided in three level as known: Data-input level, Intermediate checksum level and Seed level, they are sized as follows:

- Data level. It is composed by a 4.088x32 bit memory, 128 Kbits memory**.** As shown in Figure 6.8, it covers from row 1 up to row 4.088, marked in white colour.

- Checksum level. The protection of the 130.816 data bits with 16 frames implies 8.176 bits per frame. This requires at least 13 bits to protect the data level. Furthermore, the Frame architecture adds 1 parity bit per frame. This non-optimized scenario requires 2 more bits in checksum being a total of 15+1 bits required for the checksum level per frame. As shown in Figure 6.8, it covers from row 4.089 up to row 4.096. Checksum bits are marked in green color, parity bit is marked in yellow color and the extra non-used bits are marked in grey color.

- Seed level. It would be composed by 4 bits per frame as shown in Table 6.1 but it is composed by 5 bits per frame as shown in the non-optimized scenario of Figure 6.8. It covers from row 4.097 up to row 4.100, marked in purple color. This 32 bit words are not part of memory although it is represented together for simplicity.

Once architecture is explained, only one MCU event has been included and marked in red color in Figure 6.8:

- First MCU event is a skew event that causes a 6 bit MCU in words 4087 and 4088. This is within the maximum capability of Multilevel Framed HSB architecture with ID equal to 4 bits. This MCU event affects the area reserved to data. Bits affected are referenced as:
    1. Frame 10, Bit 8.172, error in Data
    2. Frame 11, Bit 8.172, error in Data
    3. Frame 12, Bit 8.172, error in Data
    4. Frame 13, Bit 8.173, error in Data
    5. Frame 14, Bit 8.173, error in Data
    6. Frame 16, Bit 8.172, error in Data

- Second MCU event is a skew event that causes a 7 bit MCU in words 4089 and 4090. This is within the maximum capability of Multilevel Framed HSB architecture with ID equal to 4 bits. This MCU event affects the area reserved to data and checksum. Bits affected are referenced as:
    1. Frame 1, Bit 5, error in Checksum level
    2. Frame 2, Bit 5, error in Checksum level
    3. Frame 3, Bit 5, error in Checksum level
    4. Frame 4, Bit 5, error in Checksum level
    5. Frame 5, Bit 6, error in Checksum level
    6. Frame 6, Bit 6, error in Checksum level
    7. Frame 8, Bit 5, error in Checksum level

A complete simulation of architecture has been performed with the previous MCU event included. The figures 6.9 to 6.12 are included as summary. Figure 6.9 shows the complete simulation of architecture of Figure 6.8. Figure 6.10 shows a zoom of the simulation of Figure 6.9. The vertical mark of oscilloscope is set at 205ns where an error is detected in frame number 11 being the bit 8.172 of this frame, the one that is corrected in this clock cycle. Other adjacent frames are also detected and corrected as frame 10 and 12. These bits correspond to bits placed in data.

Figure 6.11 shows a zoom of the simulation of Figure 6.9. The vertical mark of oscilloscope is set at 65ns where an error is detected in frame number 4 being the bit 5 of this frame, the one that is corrected. Other adjacent frames are also detected and corrected as frame 2 and 3. These bits correspond to bits placed in checksum level.

Figure 6.12 shows a zoom of the simulation of Figure 6.9. It shows how signal status becomes zero because the complete memory, the last 16 frames has been checked without any error. This signal becomes ‘1’ when an error is detected and keeps in this status up to the complete memory is checked without error.

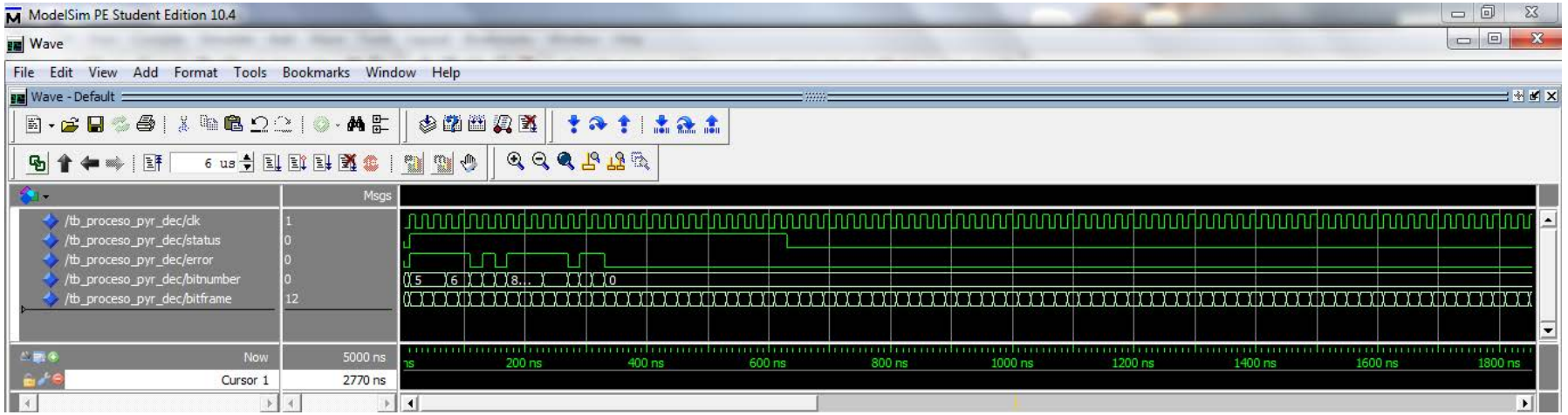


Figure 6.9. Complete simulation of Multilevel Framed HSB architecture with a distance equal to 4

Figure 6.10. First frames detected in failure in data-input level. Mark shows a SEU in Frame 11 in bit 8172 at 205ns.

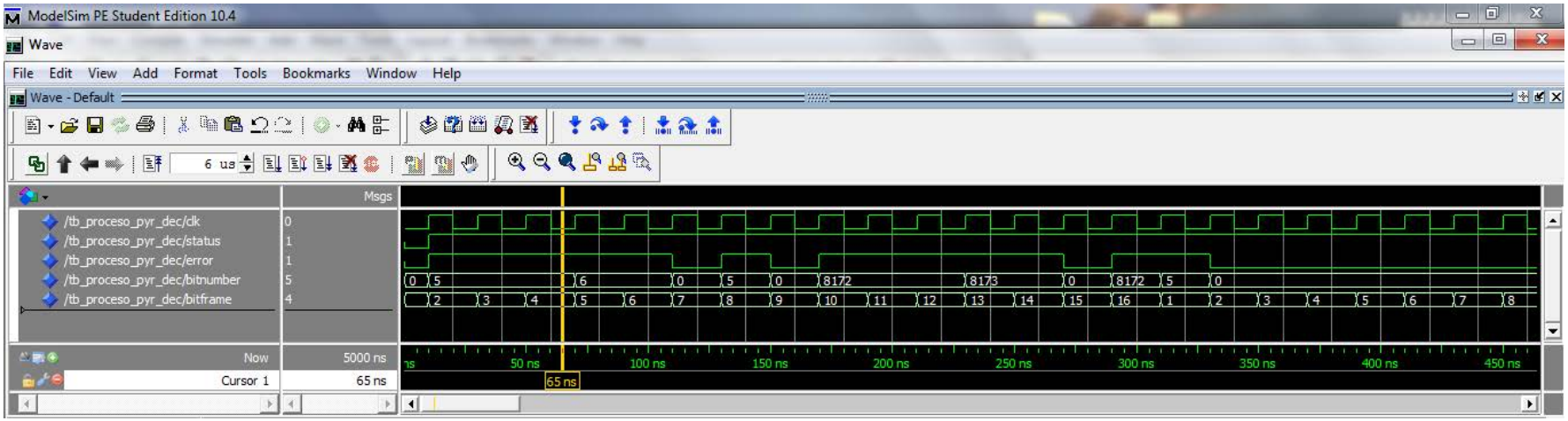


Figure 6.11. First frames detected in failure in intermediate checksum level. Mark shows a SEU in Frame 4 in bit 5 at 65ns

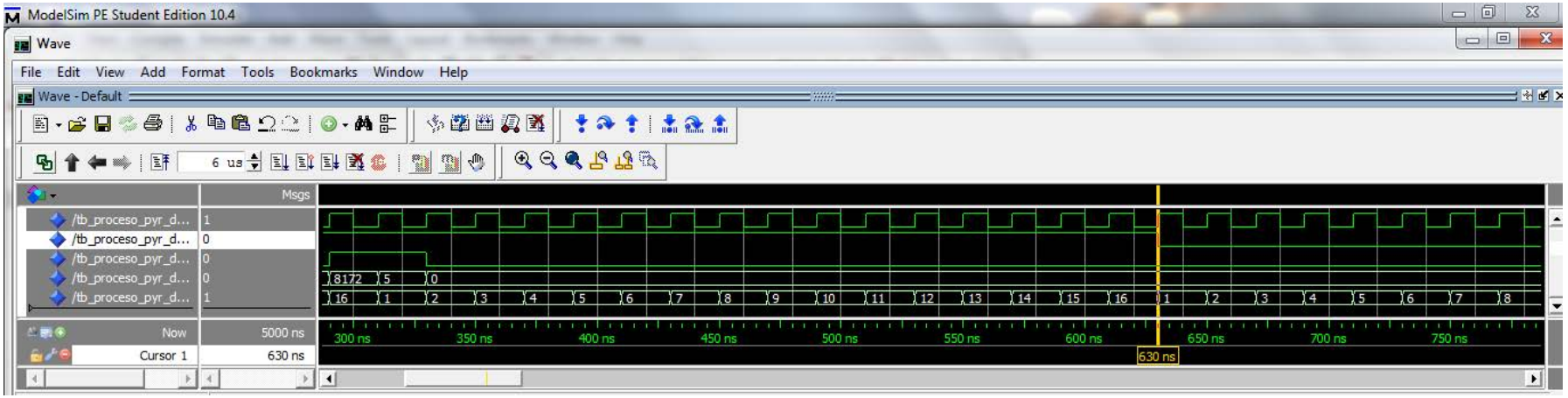


Figure 6.12. Status signal is removed once all frames have been detected without error at 630ns

6.1.3 **SCENARIO 3**

Third simulation corresponds to 248 words of 32 bits each, 8Kbit memory, managed by the Multilevel Framed HSB architecture with an **ID equal to 8bit**. Figure 6.13 represents a subset of the complete memory. The complete memory is composed by the data memory and also 16 words of 32 bits each added for checksum level that also includes the parity bits. The Framed HSB architecture also implements externally 8 words of 32 bits each for HSB protection.

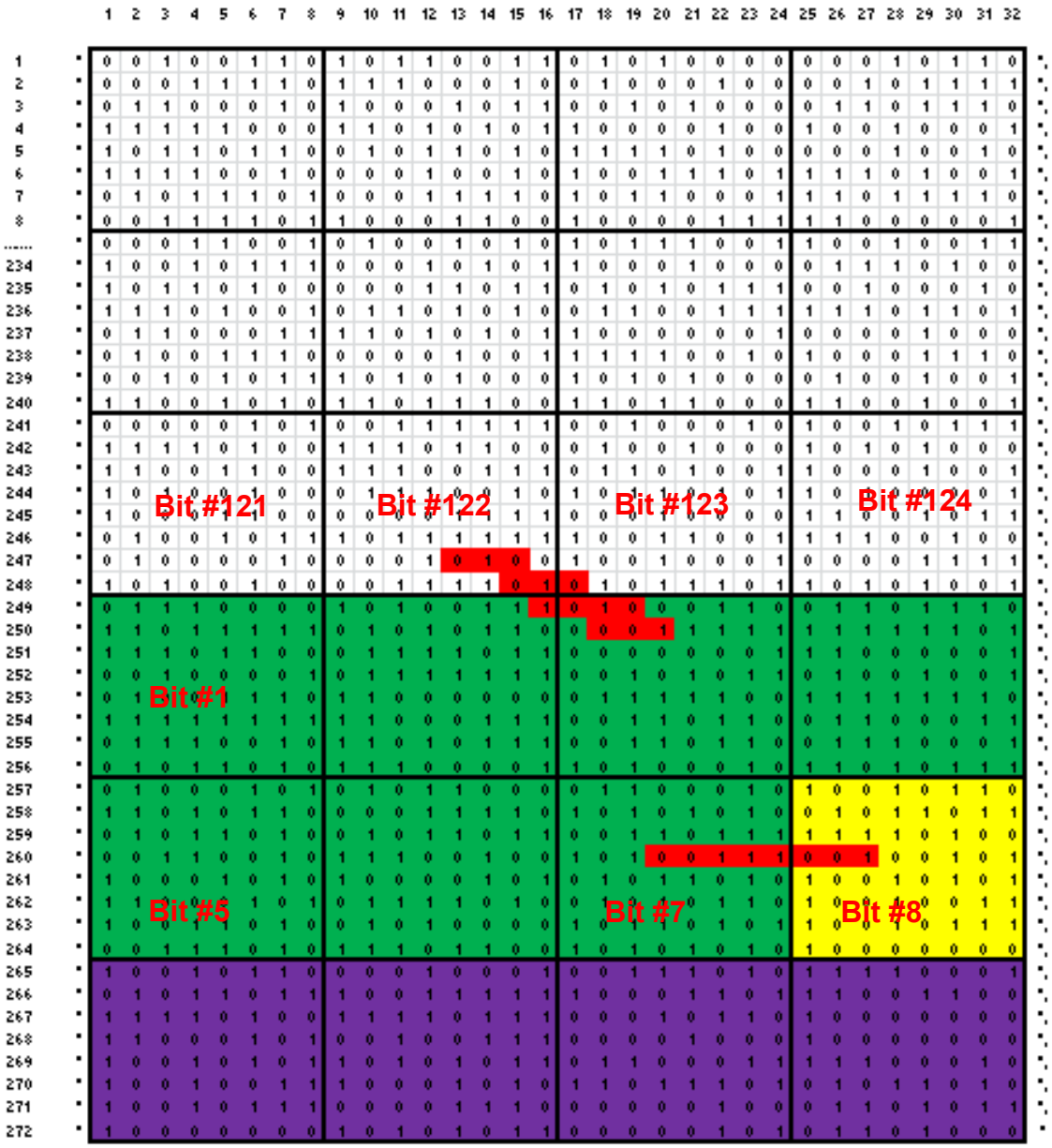


Figure 6.13. 8Kbit memory masked by
Multilevel Framed HSB architecture with distance set to 8

In this scenario, the complete memory is divided in three levels sized as:

- Data level. It is composed by a 248x32 bit memory, 8 Kbits memory**.** As shown in Figure 6.13, it covers from row 1 up to row 248, marked in white colour.

- Checksum level. The protection of the 7.936 data bits with 64 frames implies 124 bits per frame. This requires at least 7 bits to protect the data level. Furthermore, the Frame architecture adds 1 parity bit per frame. Therefore, a total of 8 bits are required for the checksum level per frame. As shown in Figure 6.13, it covers from row 249 up to row 264. Checksum bits are marked in green color and parity bit is marked in yellow color.

- Seed level. It is composed by 4 bits per frame as shown in Table 6.1. They are required to protect the 8 bits in the checksum level per frame. It covers from row 265 up to row 272, marked in purple color. This 32 bit words are not part of memory although it is represented together for simplicity.

Once architecture is explained, only one MCU event has been included and marked in red color in Figure 6.8:

- First MCU event is a pure horizontal event that causes a 8 bit upsets in the row 260. This is the maximum capability of Multilevel Framed HSB architecture. This MCU event affects the area reserved to the checksum and also parity. Bits affected are referenced as follows:
    1. Frame 25, Bit 8, error in Parity
    2. Frame 26, Bit 8, error in Parity
    3. Frame 27, Bit 8, error in Parity
    4. Frame 28, Bit 7, error in Checksum
    5. Frame 29, Bit 7, error in Checksum
    6. Frame 30, Bit 7, error in Checksum
    7. Frame 31, Bit 7, error in Checksum
    8. Frame 32, Bit 7, error in Checksum

- Second MCU event is a skew event that causes a 13 bit MCU. This is within the maximum capability of Multilevel Framed HSB architecture with ID equal to 8 bits. This MCU event affects the area reserved to data and checksum. Bits affected are referenced as:
    1. Frame 1, Bit 3, error in Checksum
    2. Frame 2, Bit 3, error in Checksum

3. Frame 3, Bit 3, error in Checksum
4. Frame 8, Bit 2, error in Checksum
5. Frame 11, Bit 3, error in Checksum
6. Frame 12, Bit 3, error in Checksum
7. Frame 53, Bit 122, error in Data
8. Frame 54, Bit 122, error in Data
9. Frame 55, Bit 122, error in Data
10. Frame 57, Bit 123, error in Data
11. Frame 63, Bit 122, error in Data
12. Frame 64, Bit 122, error in Data
13. Frame 13, Bit 3, error in Checksum

A complete simulation of architecture of Figure 6.13 is shown in Figure 6.14 Figure 6.15 represents a zoom of the Figure 6.14. The vertical mark of oscilloscope is set **at 164ns** where an error is detected in frame number 54 being the bit 122 of this frame, the one that is corrected in this clock cycle. Other adjacent frames are also detected and corrected as frame 53 and 55. These bits correspond to bits placed in data.

Figure 6.16 represents a zoom of the Figure 6.14. The vertical mark of oscilloscope is set **at 885ns** where an error is detected in frame number 26 being the bit 8 of this frame, the one that is corrected. Other adjacent frames are also detected and corrected as frame 25 and 27. These bits correspond to bits placed in parity bits. Furthermore, there are other frames 28, 29, 30, 31 and 32, where an error is detected other bit section that corresponds to bits 7, that corresponds to bits placed in checksum.

Figure 6.17 shows a zoom of the simulation of Figure 6.4. It shows how signal status becomes zero because the complete memory, the last 64 frames has been checked without any error. This signal becomes ‘1’ when an error is detected and keeps in this status up to the complete memory is checked without error.

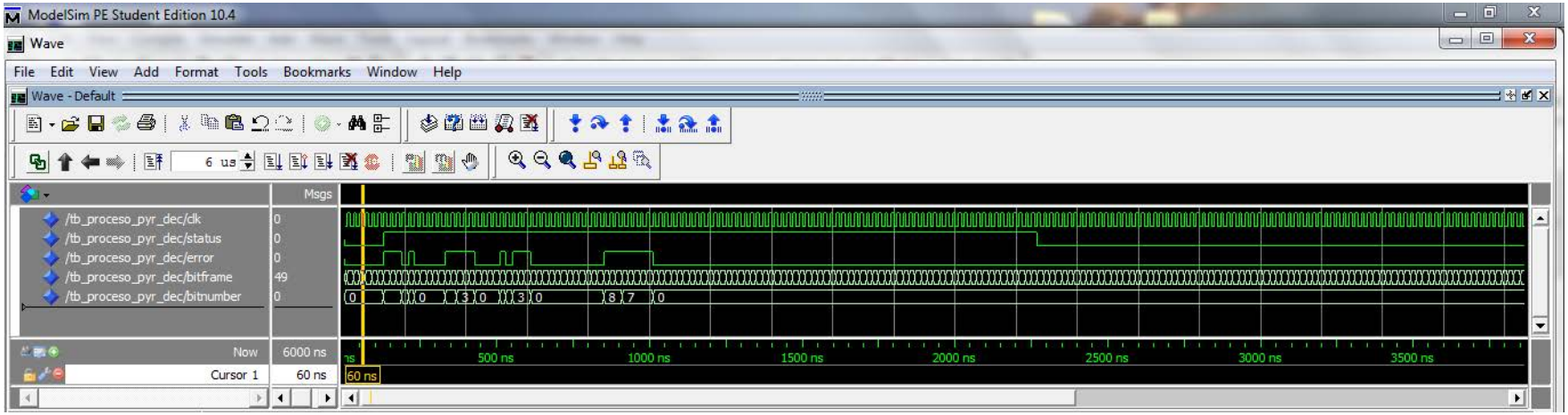


Figure 6.14. Complete simulation of Multilevel Framed HSB architecture with a distance equal to 8. 8Kbits

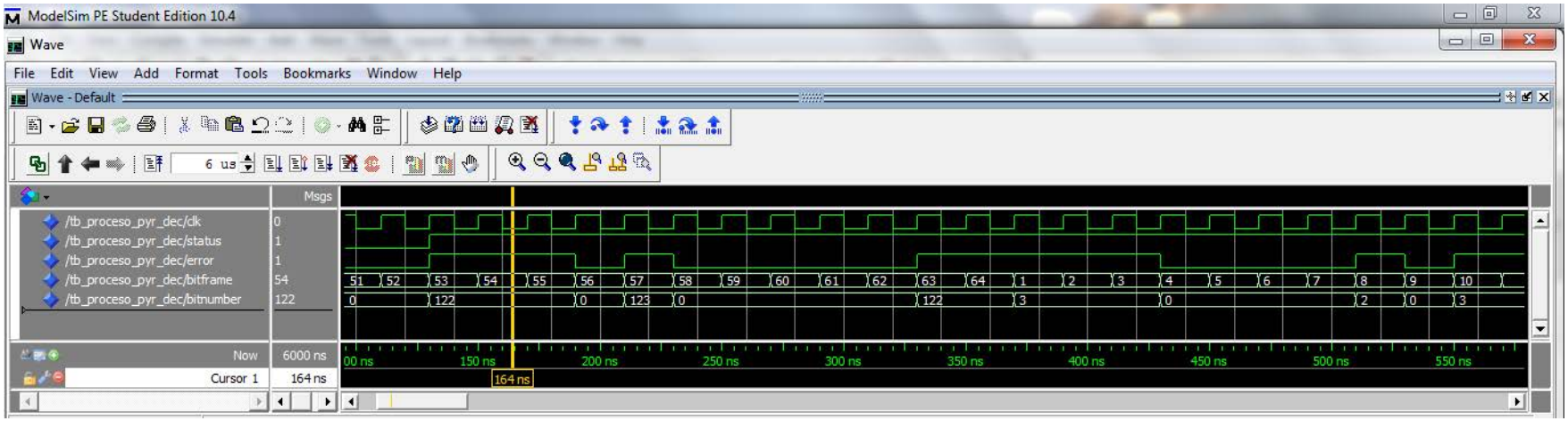


Figure 6.15. First frames detected in failure in data-input level. Mark shows a SEU in Frame 54 in bit 122 at 164ns.

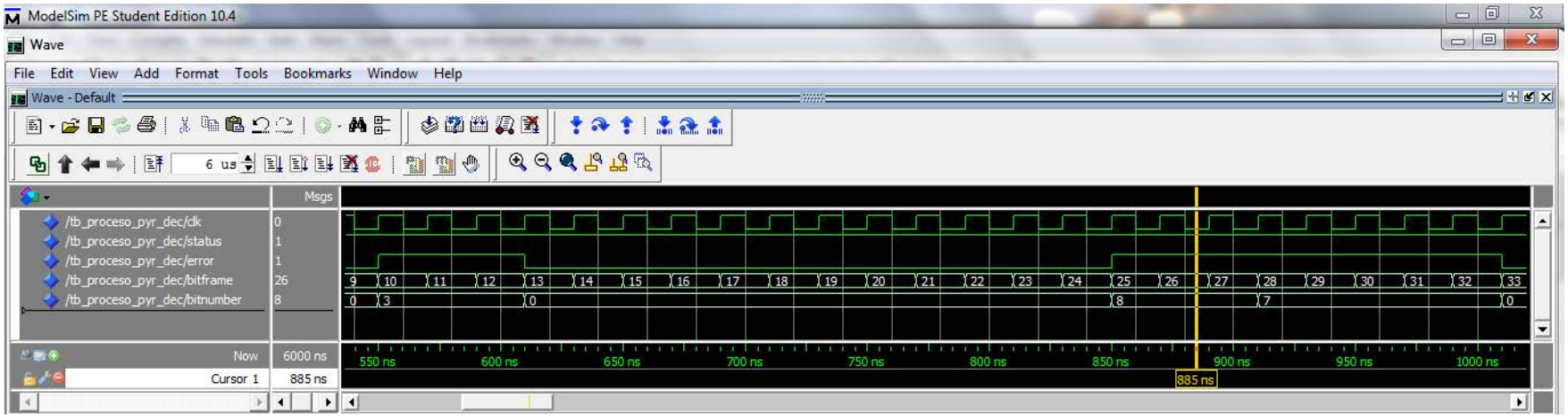


Figure 6.16. First frames detected in failure in parity. Mark shows a SEU in Frame 26 in bit 8 at 885ns.

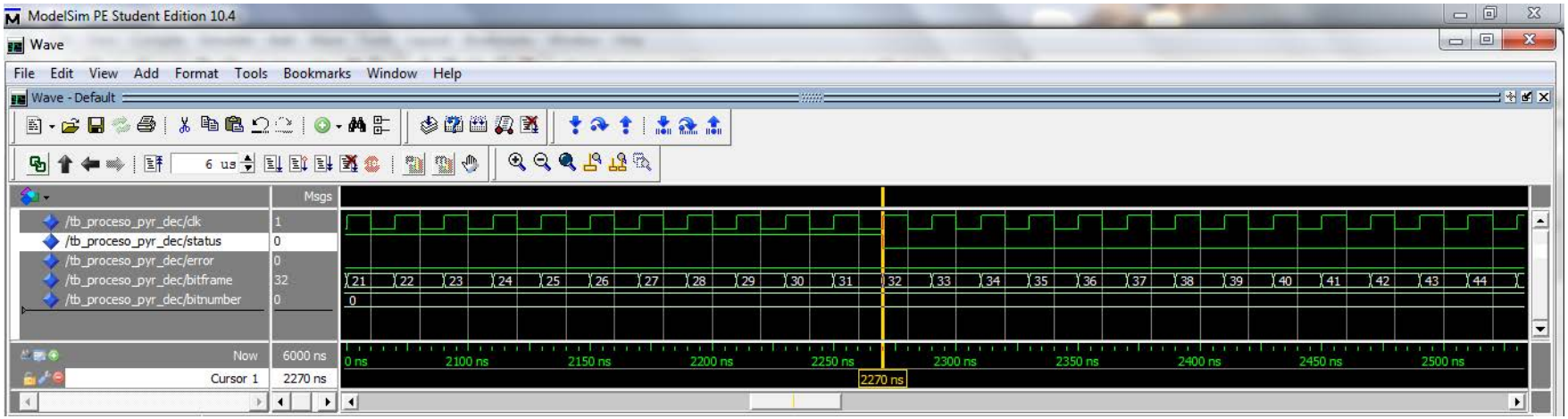


Figure 6.17. Status signal is removed once all frames have been detected without error at 2270ns

6.2 EXPERIMENTAL RESULTS

Fault-tolerant HSB architecture development has been briefly presented above. Some examples of this architecture have been implemented, simulated in VHDL for different memory sizes, then synthetized and placed & routed into a Xilinx Spartan FPGA to check the feasibility of the system using the Xilinx ISE software. Results obtained depend quantitatively on the type of device in which the algorithm is implemented, although it enables having an example in terms of time and logic consumption for a qualitative assessment.

In order to evaluate the results obtained of Frame HSB architecture under different scenarios (normal operation and error injection operation), different techniques have been also implemented for comparison purpose. The results obtained, related to the timings and resources, after the synthesis and placed & routed of them into a Xilinx Spartan FPGA have been used to compare the techniques.

BCH and Hamming codes were considered for assessing the proposed Framed HSB architecture. BCH code is a natural multiple-error correcting code. It is valid for MCU purposes and there is no need for interleaving a scheme in it. Reed-Salomon code (RS) was also studied, being recognized to be a special case of multilevel BCH codes, in which decoding methods are similar to those used for binary BCH, [6], [7].

Hamming code has been implemented although it is a single-error correcting code that is not valid for MCU purposes. For that reason, Hamming code has been also implemented with an interleaving distance technique, similar to the one used in Frame HSB architecture, in order to be able to overcome MCU events.

Therefore, the comparison performed, in terms of timing and resources, between techniques are defined as follows:

- BCH coding designed for two bit error correction capability.
- Framed Hamming coding with 64 frames, for 64 bit error correction capability.
- Multi-level Framed HSB architecture with 64 frames, for 64 bit error correction capability.

According to Framed HSB architecture error correction capabilities, BCH architecture with a 64 error-correction capability should be implemented for a fair comparison. However, the implemented BCH architecture in the comparison

is able to correct only up to two errors. This is because the simplest BCH (two errors) already implies a significant amount of area resources (adders and multiplexors), which means that an actual 64-bit BCH would perform worse (more time and resources demand) than the 2-bit BCH considered in the example.

All the examples considered in this section were implemented under the FT approach. As commented in paragraph 4.2, Fault-tolerant systems are able to continue the normal operation even in the case of the failure in its components. Redundancy techniques include provisions of extra functional capabilities that would be unnecessary in a fault-free environment. For this reason, Hamming and BCH functionality has been triplicated, TMR basis, for a fault tolerant system, whereas the proposed HSB architecture is simply duplicated.

Three scenarios with errors have been also considered: 0, 2, and 8 errors respectively. While Hamming and BCH will take the same time whatever the number of errors is injected, HSB architecture will spend less time as fewer errors are injected, because HSB architecture dynamically performs the detection/correction decoupling.

Furthermore, Xilinx Spartan FPGA was used to compute these bitstreams with errors. These manually modified bitstreams were used in Xilinx Spartan FPGA as a fault generator to demonstrate the retrieval capabilities of HSB architecture against MCU events. Furthermore, six different static memory data patterns were also used in those tests as a base of the static memory data with errors injected [39]: all-ones (AO), all-zeros (AZ), checkerboard (CB), column-stripe (CS), row-stripe (RS) and random board (RB) patterns.

Table 6.3 details the results obtained from the implementation of the different techniques over a 128 Kbit memory. This memory size is set only as an example. Actually, this architecture can be easily scaled by modifying the number of HSB per frame.

As shown in table 6.3, main drawback that makes BCH the heaviest technique in terms of computational resources needed (adders and multiplexors), in algebraic or even for Euclid method. Framed HSB architecture reduces the resources required due to the timing sequential procedure for error detection and correction. Furthermore, the error-detection computation time in Frame HSB architecture in one frame is lower than 14ns that is 7-8 times lower than BCH, improving the cycle occupation performances of systems working between 10 to 100 KHz. In Framed Hamming coding, each frame takes 67ns that is almost 5 times higher than Framed HSB architecture.

Table 6.3 distinguishes between the cycle time and the total time. One frame scrubbing requires one clock cycle time. The whole memory scrubbing requires 64 frames in these examples and, therefore, the total time requires 64 cycle times. In the Framed HSB architecture, since total time depends on the number of errors (Equation 4.11 due to detection/correction decoupling), cycle time is obtained as an average value of total time.

		Av. one cycle (ns)	Total timing (ns)	Mux	Adders	Cycles
BCH (2 error)		110,7	110,7	4e6	4e5	1
Framed Hamming		66,9	4281,6	1.680	6	64
HSB	0 error	13,89	889,5	1.061	4	64
	2 error	14,65	938,1	1.061	4	64
	8 error	16,93	1083,9	1.061	4	64

Table 6.3. Detailed timing comparison between techniques for 128 Kbits.
Timings expressed in nanosec (ns).

Best time performance is carried out by BCH, 110,7 ns. cycle time and total time are the same since no interleaving frame approach has to be considered. However, BCH is not very viable since it demands heavy computational implementation (as seen in Fig. 6.18 to 6.22).

The total time of the HSB architecture is lower (better) for all error scenarios when compared with Frame Assisted Hamming. Furthermore, the frame sweep sequential procedure of the HSB architecture eases the insertion of the SEC execution in a clock cycle. This might be interesting in synchronous systems working at a fixed frequency, where the available window of time to compute all intended operations is restricted. For instance, in Table 6.3, the average operation cycle in HSB architecture is around 15 ns, whereas it is more than 100 ns at BCH. This means that HSB adds a degree of freedom with respect of BCH to give way to the intended operations, without lowering the clock cycle frequency.

As commented in Chapter 2, different techniques are used to detect or correct errors as: Parity bits (PAR), Double Modular Redundancy (DMR), Triple Modular Redundancy (TMR), Multiple Error Correction techniques (MEC) as BCH, R-S, Berger. They are included in the Figure 6.18 for comparison. Framed HSB architecture (FHSB) has been added to Figure 6.18 and also the computation time for just one frame (HSB). Furthermore, it is also added the Framed Hamming architecture (FHam) implementation and also the computation time for just one frame (Ham).

Optimum solution should overcome MCU events, having the lowest scrubbing time and also the minimum resources. For that reason, green area (optimum resource/timing ratio) is marked taking into account that redundant resources should be integrated in same memory size (<1unit) within a timing constraint below 1us, typical functionality time requested in systems working between 10 to 100KHz.

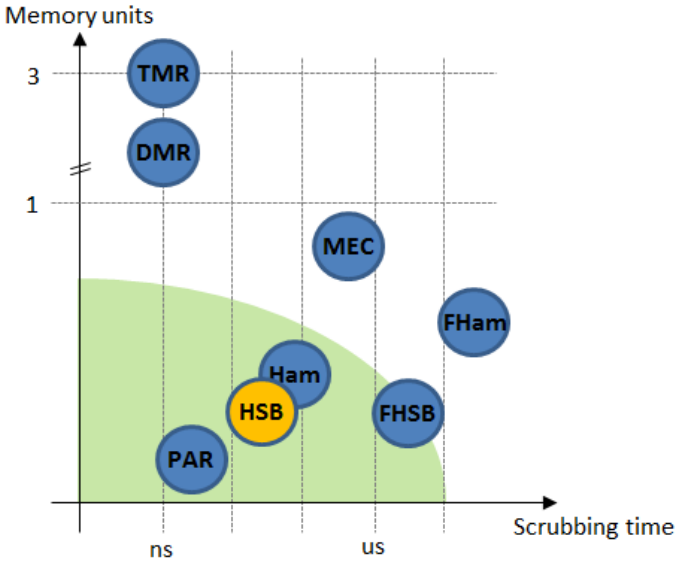


Figure 6.18. State-of-the-art with novel Framed HSB architecture.

6.2.1 TIMING ANALYSIS BASED ON EXPERIMENTAL RESULTS

As commented before, in order to evaluate the experimental results obtained in Frame HSB architecture, different techniques (Hamming and BCH) have been also implemented for comparison purpose, mainly timing of operation and also resources required for implementation.

As explained in paragraph 4.3, this capability to separate in two procedures allows launching the error-correction procedure (A in Eq. 6.1) only when an error is previously detected by the error-detection (B in Eq. 6.1). Thanks to this, the computation time is hugely reduced in Framed HSB and timing spent to compute the complete memory depends on numbers of error detected in frames. Equation 6.1 rules the timing spent to compute whole memory depends on number of frames corrupted, x, always lower than number of frames, ID^2:

$$Timing = A \bullet x + B \bullet ID^2 \qquad \text{Eq. 6.1}$$

Table 6.4 represents the results obtained in the operational timing for Framed HSB architecture. In a 128KBits memory, detection time (B) is at least three times lower than the correction time (A). This implies a significant

reduction in the operational timings of architecture because the correction procedure will be only launched once an error has been previously detected in the frame. Therefore, according to SEU/MCU rate in airborne electronics, most of the scrubbing operations, there is no error found in memory.

Table 6.4 includes the results obtained for different memory sizes where different examples of timings are also included for different number of errors injected in the complete memory. It is important to highlight that there is a column included for 12 errors because this is the maximum range of possible bits affected statistically by a unique MCU event.

Memory bits (Kbits)	Bits per frame 64 frames	y = Ax + 64B A	B	No error	1 error	2 error	12 error	64 error
8	124	11,9	4,82	308,8	320,7	332,6	451,6	1070,4
16	252	13,8	5,59	358,1	371,9	385,7	523,7	1241,3
32	508	14,3	5,95	380,8	395,1	409,4	552,4	1296,0
64	1.020	17,7	6,68	427,8	445,5	463,2	640,2	1560,6
128	**2.044**	**19,1**	**6,95**	**444,8**	**463,9**	**483,0**	**674,0**	**1667,2**
256	4.092	21,55	7,84	501,8	523,3	544,9	760,4	1880,9
512	8.188	24,31	8,85	566,1	590,4	614,7	857,8	2122,0

Table 6.4. Detail timing comparison. Timings expressed in nanosecond (ns).

The timing analysis shown in Table 6.4 reveals that although both functionalities, Framed HSB and Hamming, are quite similar in construction with Framed HSB architecture, main advantage of Framed HSB is the capability to separate the detection procedure and the correction procedure because most of the EDAC techniques, as Hamming, only uses correction means.

Table 6.4 shows some important results when compared memory sizes and number of error induced in memories. Taking as an example the memory of 512Kbits, following important timing improvements are highlighted as:

- Time spent to compute the memory could vary in steps of 24ns depending of number of errors in an MCU event. In case of no error, timing is 566 ns and timing increases up to 614 ns when there are two frames affected by the unique MCU event.

- According to normal low neutron fluxes at A/C altitudes, most of times there is not errors in memory. This Frame architecture corrects error when they have been detected before. In case of no error in memory, this architecture needs only 566 ns to check the complete memory while other techniques, as Hamming, require computing the full correction procedure, 2122 ns. That is 3,8 times quicker than any other technique.

- According to MCU event probabilities as explained in Chapter 2, maximum number of bits affected in an unique MCU event is lower than 12 bits. This takes 858 ns in the Framed HSB architecture because only corrects 12 frames as worse MCU event. It means 2,5 times quicker than the complete correction, 2122 ns.

Figure 6.19 is an extension of Table 6.4. It depicts a time consumption comparison in a memory size sweep up to 512Kbits. All conclusions previously explained for Table 6.4 are also applicable here where an increase of the scrubbing time versus the memory size for all cases are found.

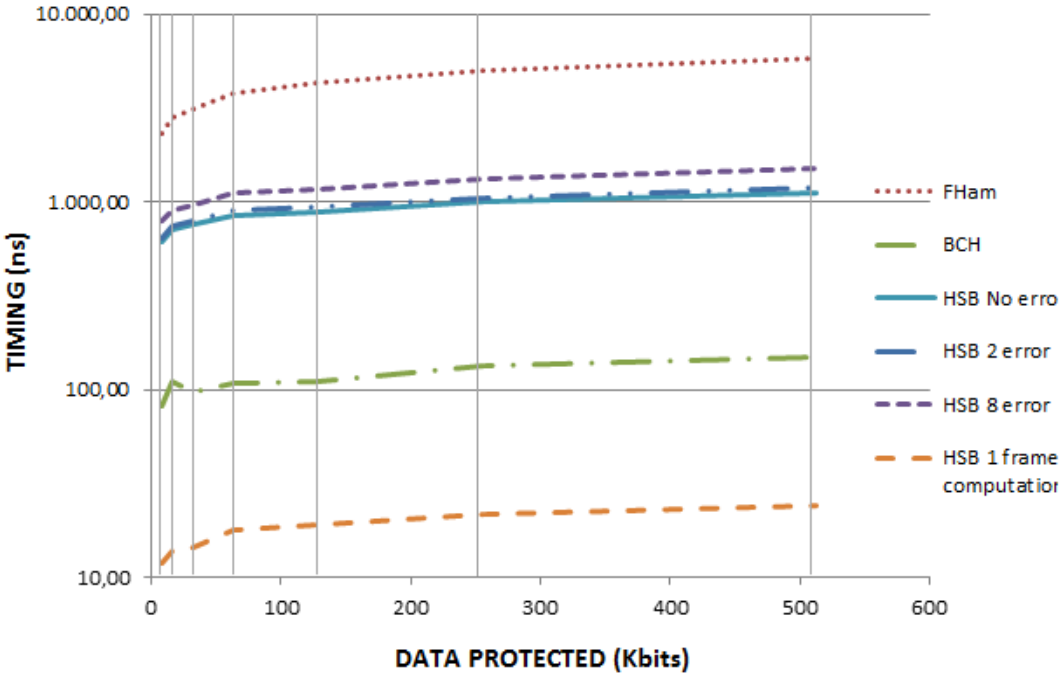


Figure 6.19. Timing consumption up to 512Kbits

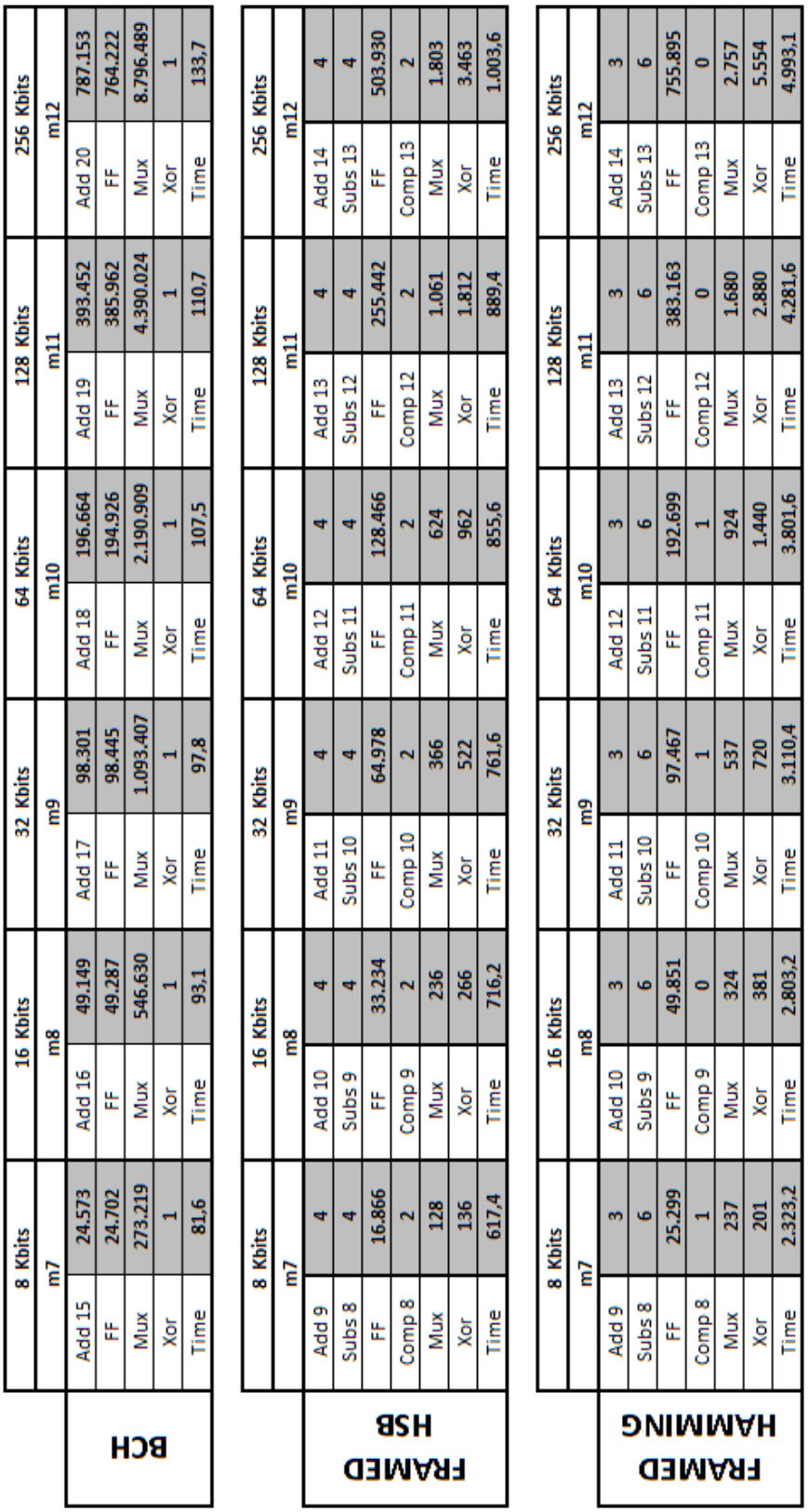
6.2.2 RESOURCE ANALYSIS BASED ON EXPERIMENTAL RESULTS

The evaluation of the experimental results in terms of resources (FFs, adders, mux) in Frame HSB architecture is one of the key characteristics of architecture to achieve the Fault-Tolerant capability. As higher resources required for implementing the architecture, the failure probability due to SEU/MCU event will also increase, as expressed in Eq. 5.2. This evaluation has been performed by the comparison with other techniques as Hamming and BCH. Table 6.5 includes the results obtained from the Synthesis and Place&Route performed with the Xilinx ISE platform in the three architectures.

Furthermore, different memory sizes are used for this comparison from 8Kbits to 256 Kbits.

Table 6.5 also includes the comparison of the results obtained from the Synthesis and Place&Route. Different aspect of results obtained is included:

- Number of **adders (add)** used by the configuration, where is also indicated the number of the bits of the adders used.
- Number of **substractors (subs)** used by the configuration, where is also indicated the number of the bits of the substractors used.
- Number of **Flip-Flops (FF)** used by the configuration.
- Number of **comparators (comp)** used by the configuration, where is also indicated the number of the bits of the comparators used.
- Number of **Multiplexors (mux)** used by the configuration.
- Number of XOR gates **(xor)** used by the configuration.
- Timing spent expressed in nanosecond (ns)



BCH	8 Kbits		16 Kbits		32 Kbits		64 Kbits		128 Kbits		256 Kbits	
	m7		m8		m9		m10		m11		m12	
	Add 15	24.573	Add 16	49.149	Add 17	98.301	Add 18	196.664	Add 19	393.452	Add 20	787.153
	FF	24.702	FF	49.287	FF	98.445	FF	194.926	FF	385.962	FF	764.222
	Mux	273.219	Mux	546.630	Mux	1.093.407	Mux	2.190.909	Mux	4.390.024	Mux	8.796.489
	Xor	1	Xor	1	Xor	1	Xor	1	Xor	1	Xor	1
	Time	81,6	Time	93,1	Time	97,8	Time	107,5	Time	110,7	Time	133,7

FRAMED HSB	8 Kbits		16 Kbits		32 Kbits		64 Kbits		128 Kbits		256 Kbits	
	m7		m8		m9		m10		m11		m12	
	Add 9	4	Add 10	4	Add 11	4	Add 12	4	Add 13	4	Add 14	4
	Subs 8	4	Subs 9	4	Subs 10	4	Subs 11	4	Subs 12	4	Subs 13	4
	FF	16.866	FF	33.234	FF	64.978	FF	128.466	FF	255.442	FF	503.930
	Comp 8	2	Comp 9	2	Comp 10	2	Comp 11	2	Comp 12	2	Comp 13	2
	Mux	128	Mux	236	Mux	366	Mux	624	Mux	1.061	Mux	1.803
	Xor	136	Xor	266	Xor	522	Xor	962	Xor	1.812	Xor	3.463
	Time	617,4	Time	716,2	Time	761,6	Time	855,6	Time	889,4	Time	1.003,6

FRAMED HAMMING	8 Kbits		16 Kbits		32 Kbits		64 Kbits		128 Kbits		256 Kbits	
	m7		m8		m9		m10		m11		m12	
	Add 9	3	Add 10	3	Add 11	3	Add 12	3	Add 13	3	Add 14	3
	Subs 8	6	Subs 9	6	Subs 10	6	Subs 11	6	Subs 12	6	Subs 13	6
	FF	25.299	FF	49.851	FF	97.467	FF	192.699	FF	383.163	FF	755.895
	Comp 8	1	Comp 9	0	Comp 10	1	Comp 11	1	Comp 12	0	Comp 13	0
	Mux	237	Mux	324	Mux	537	Mux	924	Mux	1.680	Mux	2.757
	Xor	201	Xor	381	Xor	720	Xor	1.440	Xor	2.880	Xor	5.554
	Time	2.323,2	Time	2.803,2	Time	3.110,4	Time	3.801,6	Time	4.281,6	Time	4.993,1

Table 6.5. Results of Synthesis into a Xilinx Spartan FPGA

Fig. 6.20 to 6.22 details the logic required to deal with MCU events for the proposed Framed HSB architecture compared to the Frame Assisted Hamming and the BCH, in a memory size sweep up to 256kbits. Figure 6.20 represents the resources required in terms of Flip-flops. Figure 6.21 represents the required number of Adders while Fig. 6.22 represents the required number of Multiplexors in the FPGA.

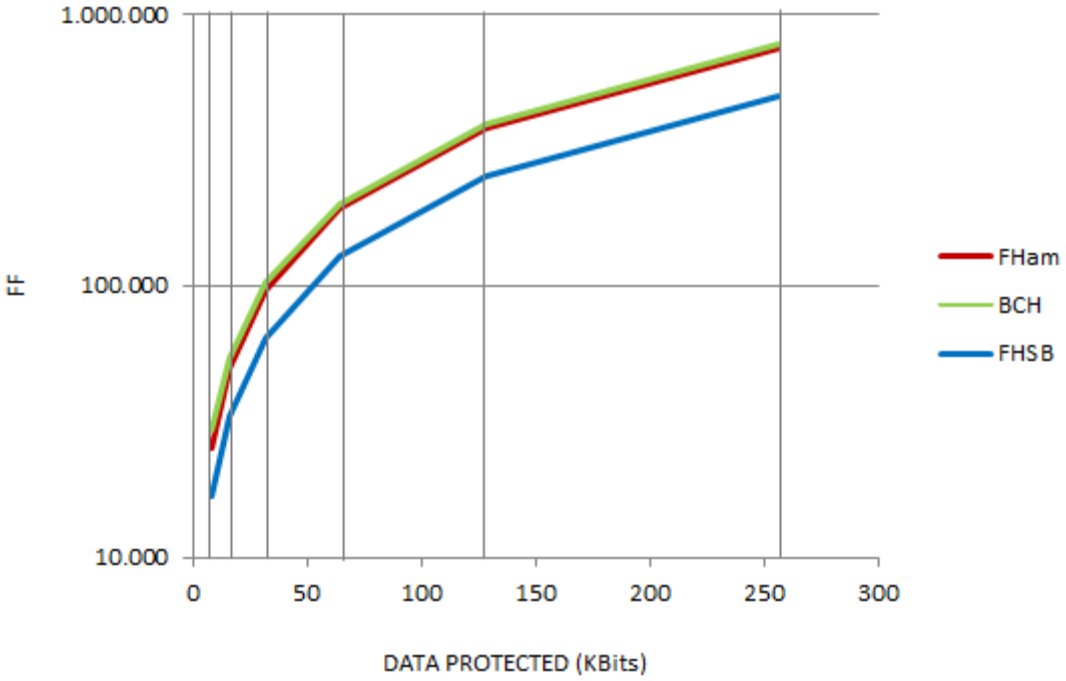


Figure 6.20. Flip-Flops architecture consumption up to 256Kbits

These comparisons also reveal that Framed HSB and Framed Hamming have similar construction, but HSB with its self-checking capability (only requires a functionality duplicate in fault-tolerant systems) permits that scrubbing function in Virtual-Framed HSB is just duplicate while Framed Hamming should be triplicate in Fault-Tolerant systems. Framed HSB uses approximately a 33% less of bitstream for scrubbing function. Resource consumption analysis also includes the comparison with BCH. In this case, figures 6.20 to 6.22 clearly show the fact that BCH decoding, based on algebraic method or Euclid method, is a heavy computational procedure that requires a huge volume of resources in FPGA.

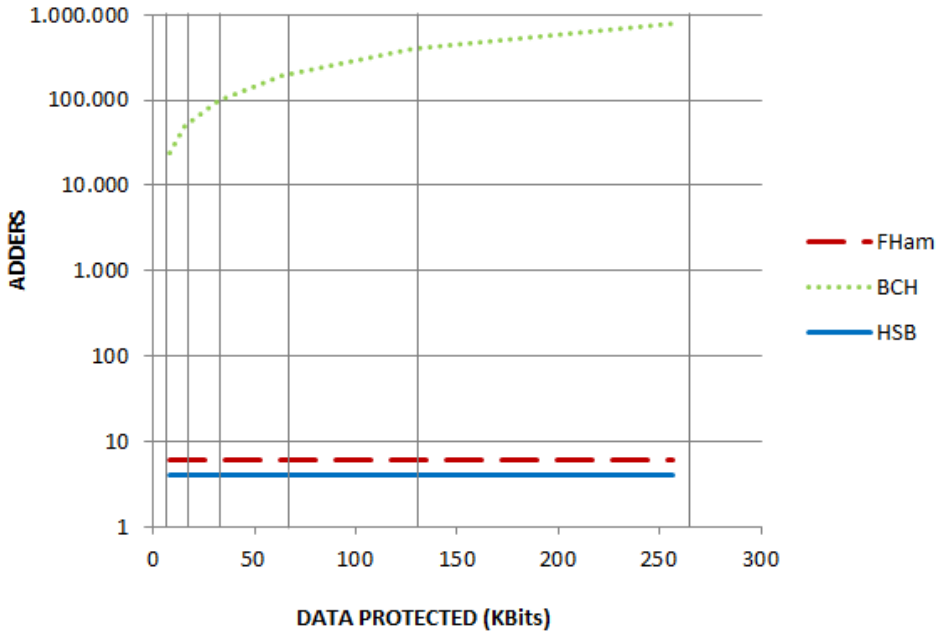


Figure 6.21. Adders consumption evolution up to 256Kbits

Odd effect shown in Figure 6.21 is that the total number of adders is kept constant even when memory size increases for Framed Hamming and Framed HSB architectures. The number of bits per adder is the characteristic that increases in same way as maximum memory capability keeping the total number of adders. In 16Kbit memory size, 10-bits adders are used in Hamming and Framed HSB while BCH uses 16-bit adders. In 256Kbits size, Hamming and Framed HSB uses 14-bits adders while BCH uses 20-bit adders. In BCH architecture, the number of adders used increases also as memory size increases. As shown clearly in Figure 6.21. BCH architecture requires at least up to three orders of magnitude more than the other two architectures in terms of Adders.

In Figure 6.22, it is represented the number of multiplexors used in each architecture. As shown clearly, BCH architecture requires at least up to three orders of magnitude more than the other two architectures. Resources required by the computation of BCH coding imply a huge number of Adders, FF and Multiplexors when compared with other techniques. Framed architectures, Hamming and Multilevel HSB, demands almost the same number of multiplexor when synthetized being the lowest one the number of multiplexor required by the HSB architecture.

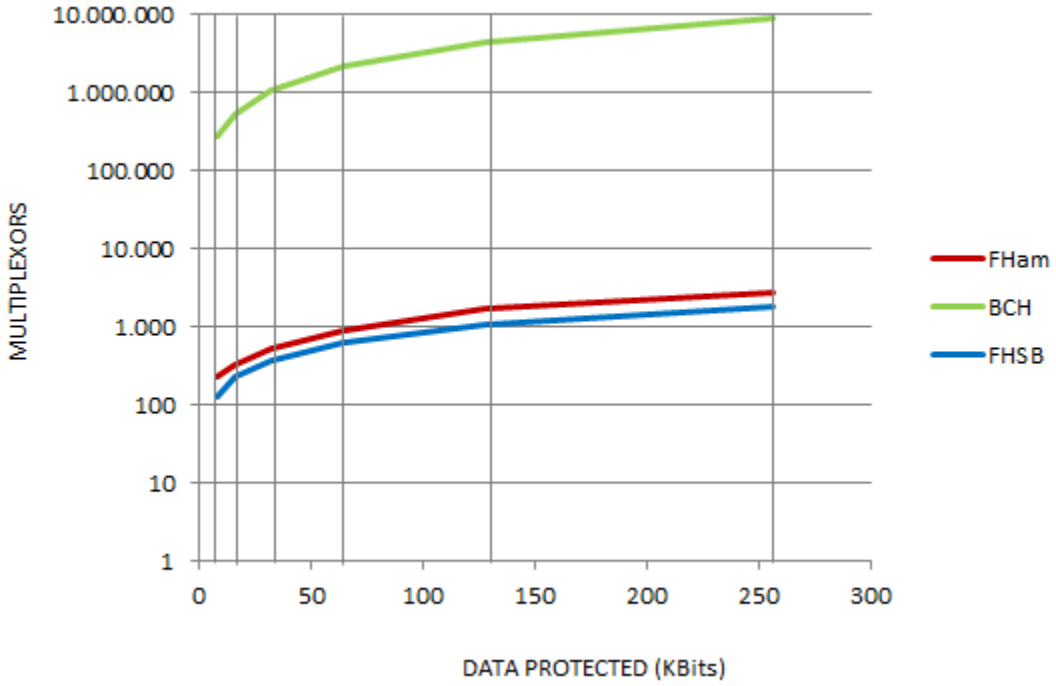


Figure 6.22. Multiplexors consumption evolution up to 256Kbits

Finally, the timing and also the resource analysis based on experimental results points several characteristics of Frame HSB architecture:

- The proposed HSB Architecture requires less area overhead than the traditional schemes.
- The lowest time consumption for the complete memory takes place under the BCH implementation that is almost 5 times lower than Framed HSB.
- The lowest cycle time consumption of the memory takes place under the Framed HSB implementation, about 1 order of magnitude less than BCH.
- BCH is not really implementable from a practical point of view, which makes HSB a reasonable solution, optimizing time and resource consumption in a balanced way.
- An additional advantage of HSB is the degree of freedom offered from the clock cycling point of view.

CHAPTER 7: CONCLUSIONS

A new Architecture to retrieve MCUs in programmable memories in a radiated environment is presented. It uses the advantages of both the reconfiguration-based techniques and redundant-based techniques to obtain a hybrid system suitable for fault-tolerant requirements, which is optimized in terms of operating time and area.

This architecture merges different well-known techniques such as:

- Single Error Correction Techniques, [5] and [6].
- Interleaving Distance concept (for instance, in [35], [36]).
- Detection and correction decoupling concept (for instance, in [7], [26]), based on an independent parity check.
- Redundancy Techniques for Error correction, [7].

It also proposes new features as:

- The implementation of a permanent reference (not vulnerable to radiation) called Hardwired Seed Bits (HSB) that contains the essential information for error retrieval. The HSB are composed of an extremely low number of bits from which the whole **memory can be reconstructed, including the EDAC functions**, under the multilevel approach of the proposed architecture.

- This system can be physically implemented as a small code at the static memory, which is transparent for the logic layer. For instance, in an FPGA, it can be implemented at the SRAM level. The content of HSB and Checkword is generated at the development phase of equipment and depends on the final code that the system protects.

- A **suitable algorithm for Fault-Tolerant application** is also included in the system in order to drive the architecture operation. It runs the operating sequence within the main program, resulting in a transparent performance for the user.

The proposed architecture allows an area reduction when compared with a TMR-EDAC approach, meaning a smaller FPGA can be selected for the system. In the example presented in this work, the proposed system performs up to 3 times quicker compared with a classical Hamming code.

7.1 SUMMARY OF ORIGINAL CONTRIBUTIONS

Proposed solution is based on a Reconfiguration-based solution that aims to restore as soon as possible the original values.

Main advantage of this kind of linear codes is the light-weighted correction coding because the calculation of each checksum bit is based on parity computations of alternate bits of data. This codification allows detecting / correcting a limited number of errors that may occur anywhere only in the data bits, not in the own redundant bits. The main contributions of the proposed Architecture are the following:

A **Multilevel Architecture,** which allows concatenating two or more levels of checksum in which every level only protects the redundant checksum bits of previous levels. The range of this multilevel architecture is scalable to any memory size.

The whole memory information is highly compressed in a set of few bits **HSB (Hardwired Seed Bits)** which are immune to radiation. From them, the EDAC (Error Detection And Correction) codes included in the Architecture are able to completely restore the whole memory.

This Architecture allows **a significant minimization of the failure probability** in memories due to radiation. This minimization consists in not only retrieving Single Event Upsets (SEUs) but also Multiple Cells Upsets (MCUs). This low probability rate opens the possibility of fulfilling Fault Tolerant requirements due to it is able to meet safety requirements such as the ones stablished at the demanding Standards of the Aerospace Industry to avoid catastrophic failures.

This Architecture proposes significant **optimizing the resources involved**. This is particularly important since solving MCUs normally increases exponentially the complexity of the system in terms of both devices involved and operation time. The contribution in this is case us the integration of different techniques such as detection-correction decoupling and bit interleaving.

This Architecture allows **Fault-Tolerant performance**. Not only it presents a very low probability failure rate, but it also extends the protection to the EDAC functions themselves.

A procedure to determine the failure probability of the proposed Architecture has been presented. This procedure allows designing the setting of the Architecture in order to fulfill the safety requirements. However, there is still a path ahead that has been identified, since there are different aspects involved in an accurate probability estimation, which has been proposes a future work. In the present work, a worst case estimation was presented as a first approach, which was useful in order to assess the experimental results.

.

7.2 FUTURE WORK

This work aims to propose a step-ahead-concept to optimize the performance of SRAM memories working in a radiated environment. This work has allowed obtaining knowledges for the complex nuclear environment that creates the radiation issue in airborne electronics and also coding mechanisms for error protection.

Several open topics and new questions have been arisen after the work performed in this Thesis. The future work will continue the development of the Framed HSB Architecture herein described. The future work-packages are presented and briefly described below:

1. Implementation technologies:

    - HSB Implementation: A basic HSB implementation could be performed with jumper pins (a pull-up/pull-down approach). However, this would not be practical for large memories, and even impossible for dynamic functions. In these cases, a programmable HSB capability is required. Manufacturers may have the solution for this issue, as inserting some FLASH Seeds Bits (FSB) in the memory. A TMR-HSB can be implemented within the memory, losing the Hardwired approach, and taking into account the tiny bits number involved.

    - Space Condition: A physical control of the bits location is required to implement the interleaving distance. The FPGA has a complex physical structure composed basically of RAM blocks which are dynamically interconnected through routing matrices. Furthermore, each RAM block itself includes elements such as a Look-Up-Table (LUT) which are configurable. The FPGA algorithm synthesizer should be reprogrammed to implement the Framed HSB Architecture functions taking into account not only the logic but also the physical bit distribution.

2. Development to explore the limits of Framed HSB Architecture

    - Time Condition: Fault-Tolerant requirement implies downsizing the failure rate to below a threshold required by the System Safety Assessment that is related to DAL. Time condition means that the system operation has to be fast enough to neglect double consecutive radiation events that take place in same iteration cycle. This condition is not only related with the clock frequency of the system but also with the fact that the whole memory is not scrubbed at once but frame by frame.

- Probability calculations: All actual systems are susceptible to failure. Therefore, safety failure probability thresholds and calculations related to double consecutive radiation impacts in Framed HSB architecture require additional investigation. The Architecture may isolate the consecutive radiation events into a set of single radiation event even in the same cycle, if they affect to different frames. Only an issue is created when consecutive radiation events in the same scrubbing cycle affect bits in common frames. This fact improves the Architecture failure probability, though it was not taken into account in this work.

- Scrubbing algorithm: As MCU appears as a set of neighbor bits affected due to a single neutron radiation event. The algorithm of Framed HSB Architecture could be improved based on an intelligent scrubbing sequence where both redundant SEC functionalities work simultaneously. They could focus the correction procedure in the neighbor frames. This algorithm could reduce the Exposition time.

3. Detection and correction of Double consecutive radiation impacts.

- Double consecutive radiation impacts: The probabilistic calculation performed in this Thesis is based on the Total Failure Rate (TFR). TFR includes all possible cases for n-Multiple Uncorrectable Impacts and reveals that TFR is equal to the Double consecutive radiation impact rate. Therefore, the unique constraint to the failure rate under this Architecture is the double consecutive radiation event. This issue requires additional investigation based on an algorithm that could complement the detection and correction of Double consecutive radiation impacts.

CHAPTER 8: REFERENCES

REFERENCES

AUTHORS

[1] JEDEC standard: JESD89A, Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices
[2] IEC62396-1 and -5, Process management for avionics, Atmospheric radiation effects.
[3] ABD100.1.9. Airbus General Requirements for Suppliers. Electronics.
[4] S. Glasstone and A. Sesonske, “Nuclear Reactor Engineering”, Van Nostrand Company. INC.
[5] S.Lin and D.Costello, “Error control coding”, Englewood Cliffs, NJ: Prentice-Hall. 2004
[6] P. Sweeney, “Error control coding from theory to practice”, Wiley. 2004

[7] F. de Lima, L. Carro and R. Reis, “Fault-Tolerance Techniques for SRAM-based FPGAs”, Frontiers in electronic testing.

[8] SRIM2013, the stopping and Range of Ions in Matter. J.F.Ziegler and J.P. Biersack.

[9] D. G. Mavis, P. H. Eaton, and M D. Sibley: “SEE characterization and mitigation in ultra-deep submicron technologies” at IEEE, pp. 105 – 112, 2009

[10] S. Serre, S. Semikh, S. Uznanski, J. L. Autran, D. Munteanu, G. Gasiot, and P. Roche: “Geant4 Analysis of n-Si nuclear reactions from different sources of neutron and its implication on Soft-error rate” at IEEE Transactions on Nuclear Science, pp. 714 – 722, 2012.

[11] J. L. Autran, , S. Serre, S. Semikh, D. Munteanu, G. Gasiot and P. Roche: “Soft-Error rate induced by thermal and low energy neutrons in 40nm SRAMs” at IEEE Transactions on Nuclear Science, pp. 2658 – 2665, 2012.

[12] S. Wen, R. Wong, M. Romain, N. Tam: “Thermal neutron soft error rate for SRAM in the 90nm-45nm technology range” at IEEE IRPS10, pp. 1036 – 1039, 2010.

[13] A. D. Tipton, J. A. Pellish, , J. M. Hutson, R. Baumann: “Device-Orientation Effects on Multiple-bit Upset in 65nm SRAMs” at IEEE Transactions on Nuclear Science. pp. 2880 – 2885, 2008.

[14] H. Quinn, P. Graham, J. Krone, M. Caffrey, S. Rezgui: “Radiation-Induced Multi-bit upset in SRAM-based FPGAs” at IEEE Transactions on Nuclear Science, pp. 2455 – 2461, 2005

[15] F. Wrobel, J.-M. Palau, M.-C. Calvet, O. Bersillon, H. Duarte: “Simulation of nucleon-induced nuclear reactions in a simplified SRAM structure: Scaling Effects on SEU and MCU cross sections” at IEEE Transactions on Nuclear Science, pp. 1946 – 1952, 2001.

[16] T. Mérelle, F. Saigné, B. Sagnes, G. Gasiot, Ph. Roche, T. Carrière, M.-C. Palau, F. Wrobel, and J.-M. Palau: “Monte-Carlo simulation to quantify neutron-induced multiple Bit Upset in advance SRAMs” at IEEE Transactions on Nuclear Science, p. 1538 – 1544, 2005.

[17] E. Normand, K. Vranish, A. Sheets, M. Stitt, and R. Kim: “Quantifying the double-sided neutron SEU threat, from low energy (Thermal) and high energy (>10 MeV) neutrons” at IEEE Transactions on Nuclear Science, pp. 3587 – 3595, 2006.

[18] R. Baumann et al: “Radiation-Induced Soft errors in advance semiconductor technologies” at IEEE transactions on device and materials reliability, pp. 305 – 316, 2005.

[19] G. Hubert et al: “Detail analysis of secondary ions effect for the calculation of neutron induced SER in SRAMs” at IEEE Transactions on Nuclear Science, pp. 1953 – 1959, 2001.

[20] D. Brown (2011 December), National Nuclear Data Center (ENDF), [Online]. Available: www.nndc.bnl.gov
[21] C. Dyer and F. Lei: “Monte Carlo calculations of the Influence on aircraft radiation environments of structures and solar particle events” at IEEE Transactions on Nuclear Science, pp. 1987 – 1995, 2001.
[22] A. D. Tipton, X. Zhu, H. Weng: “Increased rate of multiple-bit upset from Neutrons al large angles of incidence” at IEEE Transactions on device and materials reliability, pp. 565 – 570, 2008.
[23] ShiJie Wen et al: “B10 findind and correlation to thermal neutron soft error rate sensitivity for SRAM in the sub-micron technology” at IEEE IIRW. 2010
[24] F. Wrobel, J.-M. Palau, M. C. Calvet, O. Bersillon, and H. Duarte: “Incidence of multi-particle event on soft error rates caused by n-Si nuclear reactions” at IEEE transactions on nuclear science, pp. 2580 – 2585, 2000.
[25] Microsemi white paper: “Understanding SEE in FPGA”, 2011.
[26] U. Legat, A. Biasizzo, and F. Novak: “SEU recovery mechanism for SRAM-based FPGA” at IEEE Transactions on Nuclear Science, pp. 2562 – 2571, 2012.
[27] G. Asadi, M. Baradaran: “Soft Error mitigation for SRAM-based FPGAs” 23rd IEEE VLSI Test Symposium, pp. 207 – 212, 2005.
[28] F. de Lima et al: “Single Event Upet mitigation Techniques for SRAM-based FPGAs”
[29] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, J. C. Hoe: “Multi-bit error tolerant caches using Two-dimensional error coding” at IEEE Symposium on Microarchitecture, pp. 197 – 209, 2007.
[30] S. Gregori, A. Cabrini, O. Khouri: “On-chip error correcting techniques for new-generation flash memories” at Proceedings of the IEEE, pp. 602 – 616, 2003
[31] C. Carmichael et al: “SEU mitigation technique for Virtex FPGAs in Space Applications”, Xilinx and LANL.
[32] Altera white papers, “Error correction code in SoC FPGA-based Memory system”, 2012 and “Test methodology of error detection and recovery using CRC in Altera FPGA devices”, 2011.
[33] Altera white papers (2012 April), “Error correction code in SoC FPGA-based Memory system”, Altera, [Online]. Available: https://www.altera.com/en_US/pdfs/literature
[34] B. Gill, M. Nicolaidis, C. Papachristou: “Radiation induced Single-word Multiple-bit upsets correction in SRAM” at IEEE IOLTS, pp. 266 – 271, 2005.

[35] Michael Andrew Clemens et al: “The Effects of Neutron Energy and High-Z Materials on Single Event Upsets and Multiple Cell Upsets” at IEEE TNS, 2011.
[36] David F. Heidel et al: “Single-Event Upsets and Multiple-Bit Upsets on a 45 nm SOI SRAM” at IEEE TNS, 2009.
[37] Xiaowei Zhu et al: “A Quantitative Assessment of Charge Collection Efficiency of N+ and P+ Diffusion Areas in Terrestrial Neutron Environment” at IEEE TNS, 2007.
[38] RTCA-DO178: Software Considerations in Airborne Systems and Equipment Certification
[39] RTCA-DO254: Hardware Considerations in Airborne Systems and Equipment Certification
[40] ARP4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment
[41] ARP4754A: Guidelines For Development Of Civil Aircraft and Systems
[42] RTCA-DO160: Environmental Conditions and Test Procedures for Airborne Equipment

CHAPTER 9: ADDENDA

This Addenda was developed under the demand of the Evaluation Board of the PhD Thesis entitled “Novel Fault Tolerant Multi-Bit Upset (MBU) Error-Detection and Correction (EDAC) architecture” written by Andrés Jiménez Olazábal.

This PhD Thesis was officially reviewed on the 26th of June of 2018. The members of the Evaluation Board were Luis Entrena Arrontes, Mª Luisa López Vallejo and Pedro Reviriego Vasallo. The work was evaluated as “passed”, although the Evaluation Board demanded some points to be added.

The author has created this new Chapter (Chapter 9) in his best understanding that its content fully satisfies the demanded requirements of the Evaluation Board. Chapter 9 was written and attached to the original version the 16th of July of 2018, without introducing any change at any part of the rest of the document.

Chapter 9 is composed of 3 sections. Section 9.1 lists and propose a correction for different errata founded at the original text. Section 9.2 clarifies some concepts pointed by the Evaluation Board. Section 9.3 widens the bibliography offered at the original text.

9.1 ERRATA SHEET

1. Pag 15: Work target.
   Error said "*The solution intended in this Thesis is based on an error correcting technique with a novel permanent reference not affected by SEU/MCU, called "Hardwired Seed Bits", HSB, physically soldered on the PCB*"

   Correction: "*The solution intended in this Thesis**, mainly to protect the FPGA configuration static memory, [73] and [74],** is based on an error correcting technique with a novel permanent reference not affected by SEU/MCU, called "Hardwired Seed Bits", HSB, physically soldered on the PCB*"

2. Pag 15: Work target.
   Error said *"Main aim of the solution proposed is the MCU protection. As MCU appears in bits physically near each other, a single-error correcting technique fails in the detection of MCU events"*

   Correction: *"Main aim of the solution proposed**, [73] and [74],** is the MCU protection. **Traditional conceptual matrix model for memories have been used as reference for this work. Therefore**, MCU appears in bits physically near each other, single-error correcting techniques fails in the detection of MCU events"*

3. Pag 33: Multiple Bit Upset
   Error said *"Multiple Bit Upset (MCU)"*

   Correction *"Multiple Bit Upset (**MBU**)"*

4. Pag 35: Current solutions in Airborne electronics
   Error said *"State-of-the-art of different FPGA suppliers allows the partial reconfiguration to rewrite only the selected part without stop the circuit implemented in device for a shorter period of time"*

   Correction *"State-of-the-art of different FPGA suppliers **allow** the partial reconfiguration to rewrite only the selected part without **stopping** the circuit implemented in device for a shorter period of time"*

5. Pag 36: Current solutions in Airborne electronics
   Error said *"The more traditional method of verification of the data stored in configuration memory is to readback the data and performs a bit for bit comparison. This requires the use of the original file and the readback file which are equal in size to the original bit-stream used to configure the FPGA. This method would effectively require triple the amount of system memory to detect SEU events.*

   Most popular technique is EDAC based on Single/Multiple-error correcting techniques, as Hamming, BCH, R-S, Berger codes that reduces the resources required. The most important drawback of multi-error correcting codes is that their logic complexity increase exponentially as

range of maximum error correction capability, Figure 2.17. This topic is an important factor to take into account when architecture is designed for MCU correction"

Correction *"The more traditional method of verification of the data stored in configuration memory is to readback the data and **perform** a bit for bit comparison. This requires the use of the original file and the readback file which are equal in size to the original bit-stream used to configure the FPGA. This method would effectively require triple the amount of system memory to detect SEU events.*

*Most popular technique is EDAC based on Single/Multiple-error correcting techniques, as Hamming, BCH, R-S, Berger codes that reduces the resources required. The most important drawback of multi-error correcting codes is that their logic complexity **increases** exponentially as range of maximum error correction capability, Figure 2.17. This topic is an important factor to take into account when architecture is designed for MCU correction"*

6. Pag 44: Basic operation algorithm

   Error said *"Furthermore, the Hardwired Seed Bits, HSB, are also added in the Physical implementation side as the permanent reference of architecture. This reference contains the essential information, neutral to radiation, from which errors in original data can be retrieved. There is still a risk of having an upset in the HSB, although this risk has been eliminated due to the their physical implementation. This reference is not saved physically in SRAM memory. In order to be neutral to radiation, It shall be somehow physically implemented to prevent the vulnerability to radiation, for example, pull-up resistors, ROM or a Hardened memory"*

   Correction *"Furthermore, the Hardwired Seed Bits, HSB, are also added in the Physical implementation side as the permanent reference of architecture. This reference contains the essential information, neutral to radiation, from which errors in original data can be retrieved. There is still a risk of having an upset in the HSB, although this risk has been eliminated due **to their** physical implementation. This reference is not saved physically in SRAM memory. In order to be neutral to radiation, **HSB** shall be somehow physically implemented to prevent the vulnerability to radiation, for example, pull-up resistors, ROM or a Hardened memory"*

7. Pag 46: Basic operation algorithm

   Error said *"Figure. 3.2. Both SEC functions assess Checkword from HSB"*

   Correction *"Figure. 3.2. **SEC function assesses** Checkword from HSB"*

8. Pag 46: Basic operation algorithm

   Error said *"The Checkword level is composed of the checksum bits for correction and an additional bit for a just parity error detection, which constitutes the SEC reference for the data level to be scrubbed. SEC function can be used in this case. The possible scenarios under this step are as follows"*

   Correction *"The Checkword level is composed of the checksum bits for correction and an additional bit for **just a** parity error detection, which constitutes the SEC reference for the data level to be scrubbed. SEC function can be used in this case. The possible scenarios under this step are as follows"*

9. Pag 46: Basic operation algorithm

   Error said *"The SECs functions stored in memories might be affected themselves by an MCU, which means that the Error Correction would not be fully guaranteed in this case. Therefore, the second Issue that may be arisen is"*

   Correction *"The **SEC** functions stored in memories might be affected themselves by an MCU, which means that the Error Correction would not be fully guaranteed in this case. Therefore, the second Issue that may be arisen is"*

10. Pag 55: Virtual-framed interleaving for MCU correction

    Error said *"Figure. 4.1. 17 x 8 bit memory divided in 16 frames. Bits affected by MCU are marked in red"*

    Correction *"Figure. 4.1. **16 x 8 bit memory**. Bits affected by MCU are marked in red"*

11. Pag 69: Architecture for Fault Tolerant systems

    Error said "Figure 4.8.a represents the scenario in which both SEC functionalities could be affected by only one neutron impact. This situation shall be avoided by using the configuration shown in Figure 4.8.b"

    Correction: "Figure ***4.8*** represents the scenario in which both SEC functionalities could be affected by only one neutron impact. This situation shall be avoided by using the configuration shown in Figure ***4.9"***

12. Pag 69: Architecture for Fault Tolerant systems

    Update of Figure 4.8 and Figure 4.9.

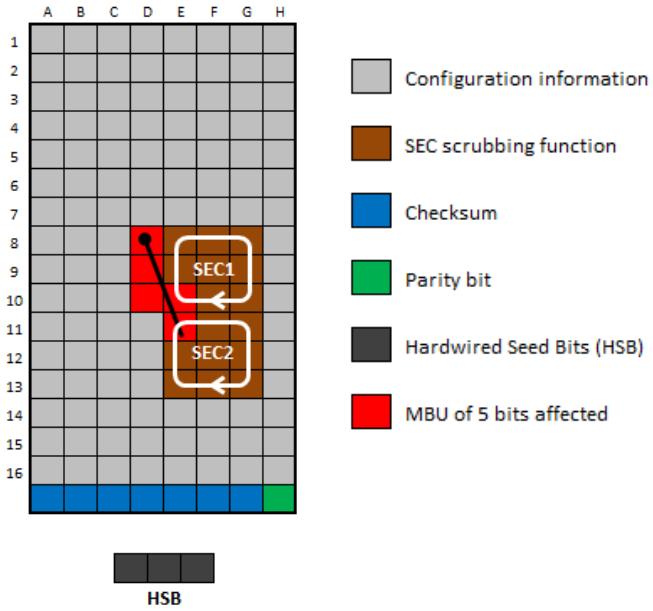


Figure 4.8. HSB architecture distribution where both duplicated scrubbing functionalities are placed in Data level. MCU event could affect to both SECs at same time. This situation shall be avoided

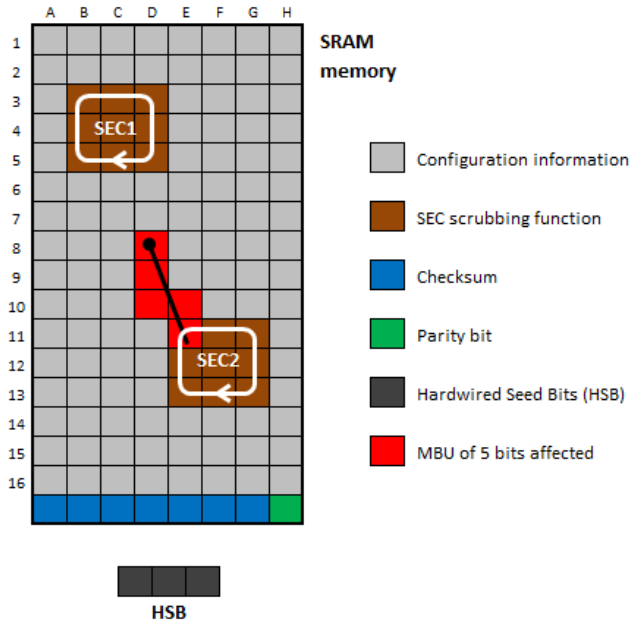


Figure 4.9. “MCU event only affects to SEC2 but no to SEC1”

13. Pag 70: Implementation issues

Error said *“All actual systems are susceptible to failure. Therefore, the Fault Tolerant definition must be stablished in terms of a failure probability threshold. Actually, in the standards stablished in the aerospace industry (ARP4754, ARP4754/A, RTCA DO254 and RTCA DO178) the severity of a system is defined in terms of DAL (Design Assurance Level). The most severe scenario is named as a catastrophic event, and the systems and equipments that may produce it in case of failure must be certified as DAL-A. According to standards, the failure probability of DAL-A systems must be no greater than $1 \cdot 10^{-9}$ 1/hr. This would means, as a rough example, that just one catastrophic failure would take place in the whole life-span of a complete fleet (1.000 aircrafts)”*

Correction *“All actual systems are susceptible to failure. Therefore, the Fault Tolerant definition must be stablished in terms of a failure probability threshold. Actually, in the standards stablished in the aerospace industry (ARP4754, ARP4754/A, RTCA DO254 and RTCA DO178) the severity of a system is defined in terms of DAL (Design Assurance Level). The most severe scenario is named as a catastrophic event, and the systems and equipments that may produce it in case of failure must be certified as DAL-A. According to standards, the failure probability of DAL-A systems must be no greater than $1 \cdot 10^{-9}$ 1/hr. This would **mean**, as a rough example, that just one catastrophic failure would take place in the whole life-span of a complete fleet (1.000 aircrafts)”*

14. Pag 79: Multilevel architecture

Error said *“Checksum code proposed in previous paragraphs is able to correct any single failure included in data bits of bitstream but no anywhere in checksum bits, because it would be wrongly understood as failure in data bits. In this way, the multilevel architecture becomes the solution for this issue. The classical solutions for mitigating the effects of SEE are divided in two main categories: reconfiguration-based (i.e. EDAC) and redundancy-based solutions (i.e. TMR).*

- *The first solutions restore the original values in memory as soon as possible after a SEE event by means of error detection and correction codes, [7], [28], [29], [30], [33].*

- *The second group masks the SEE effects, isolating the circuit outputs from internal SEUs by means of a voting process, which means that the effect of an error is not transmitted to the output signals. This scheme requires extra cost in terms of components, weight, area and power dissipation"*

Correction *"Checksum code proposed in previous paragraphs is able to correct any single failure included in data bits of bitstream but **not** anywhere in checksum bits, because it would be wrongly understood as failure in data bits. In this way, the multilevel architecture becomes the solution for this issue. The classical solutions for mitigating the effects of SEE are divided in two main categories: reconfiguration-based (i.e. EDAC) and redundancy-based solutions (i.e. TMR).*

- *The first solutions restore the original values in memory as soon as possible after a SEE event by means of error detection and correction codes, [7], [28], [29], [30], [33].*

- *The second group masks the SEE effects, isolating the circuit outputs from internal SEUs by means of a voting process, which means that the effect of an error is not transmitted to the output signals. This scheme requires extra cost in terms of components, weight, area and power dissipation, **[7], [28], [31], [70]**"*

15. Pag 80: Multilevel architecture
    Error said *"Figure 4.19. Multilevel structure. Seed is a permanent reference"*

    Correction *"Figure 4.19. Multilevel structure **for coding and decoding processes**. Seed is a permanent reference"*

16. Pag 85: Time Optimization based on Parity
    Error said *"As described in previous paragraphs, multilevel architecture is composed by three levels: Data level, checksum level and Seed level. Taking the opportunity of direct protection from the permanent reference of HSB, checksum level includes now a parity bit only of the data bits. In this case, any single failure, SEU/MCU, could be detected by just a parity computation of data-bit level"*

    Correction *"As described in **the** previous paragraphs, multilevel architecture is composed by three levels: Data level, **the** checksum level and **the** Seed level. Taking the opportunity of direct protection from the permanent reference of HSB, checksum level includes now a parity bit only of the data bits. In this case, any single failure, SEU/MCU, could be detected by just a parity computation of data-bit level"*

17. Pag 86: Time Optimization based on Parity
    Error said *"The Detection time (B in Eq 4.12) in a frame is typically faster than Correction time (A in Eq 4.12). Therefore, this technique only launches the correction procedure in a frame when an error is detected by the parity bit of frame [7]. Thanks to this, the computation time is hugely reduced because in some EDACs (as Hamming or BCH), detection & correction are coupled being correction procedure performed every time"*

Correction *"The Detection time (B in **Eq 4.11**) in a frame is typically faster than Correction time (A in **Eq 4.11**). Therefore, this technique only launches the correction procedure in a frame when an error is detected by the parity bit of frame [7]. Thanks to this, the computation time is hugely reduced because in some EDACs (as Hamming or BCH), detection & correction are coupled being correction procedure performed every time"*

18. Pag 97: Probability / Reliability calculations

Error said *"The setting of Framed Architecture with a ID non-lower than 8 bits improves the failure rate. In this scenario, the main contributor for the calculation of the failure rate is the n-Multiple Uncorrectable Impacts, n-MU-IR, where some impacts occur in same frame at same iteration cycle. Typically, the contribution of double and triple impact rates, 2-MU-IR and 3-MU-IR, would be the most important ones for the failure rate calculation. For that reason, Figure 5.4 to 5.7 also includes the probability calculations for 2-MU-IP and 3-MU-IP, needed to obtain their failure rates, 2-MU-IR and 3-MU-IR"*

Correction *"The setting of Framed Architecture with a ID non-lower than 8 bits improves the failure rate. In this scenario, the main contributor for the calculation of the failure rate is the n-Multiple Uncorrectable Impacts, n-MU-IR, where some impacts occur in same frame at same iteration cycle. Typically, the contribution of double and triple impact rates, 2-MU-IR and 3-MU-IR, would be the most important ones for the failure rate calculation. For that reason, **Tables** 5.4 to 5.7 also includes the probability calculations for 2-MU-IP and 3-MU-IP, needed to obtain their failure rates, 2-MU-IR and 3-MU-IR"*

19. Pag 98: Probability / Reliability calculations

Include Eq.5.12.

$$\mathbf{TFP} = \sum_{n=1}^{\infty} IP^n$$

Eq. 5 .12

20. Pag 99: Probability / Reliability calculations

Error said *"Figures 5.4 to 5.7 also include the probability calculations for 2-MU-IP and 3-MU-IP needed to obtain the failure rates according to the Exposition time, ExT defined for the system working at 10 KHz"*

Correction *"**Tables** 5.4 to 5.7 also include the probability calculations for 2-MU-IP and 3-MU-IP needed to obtain the failure rates according to the Exposition time, ExT defined for the system working at 10 KHz"*

21. Pag 104: Experimental Assessment

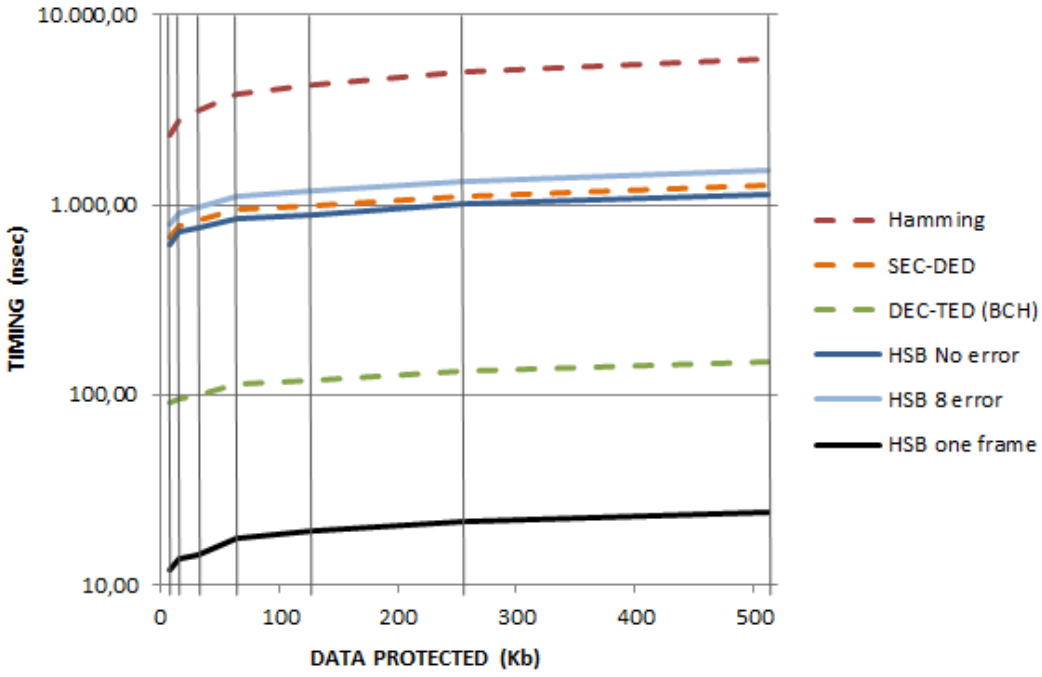
Error said *"Different simulations and implementations have been performed in the Multilevel Framed HSB architecture with different ranges of ID and HSB. This architecture has been implemented, simulated in VHDL for different memory sizes, then Synthetized and Placed & Routed into a Xilinx Spartan FPGA to check viability of system. The results obtained after the synthesis and implementation of system with redundant function logics are included in this section"*

Correction *"Different simulations and implementations have been performed in the Multilevel Framed HSB architecture with different ranges of ID and HSB. This architecture has been implemented, simulated in VHDL for different memory sizes, then Synthetized and Placed & Routed into a Xilinx **Spartan3E** FPGA to check viability of system. The results obtained after the synthesis and implementation of system with redundant function logics are included in this section"*

22. Pag 122: Scenario 3
    Error said *"Once architecture is explained, only one MCU event has been included and marked in red color in Figure 6.8"*

    Correction *"Once architecture is explained, only one MCU event has been included and marked in red color in **Figure 6.13**"*

23. Pag 131: Timing analysis based on experimental results
    Update of Figure 6.19. Timing consumption up to 512Kbits"



9.2 EXTENSION OF CONCEPTS

24. Pag 32: Single Event Upset (SEU)

"Plateau cross section determined empirically, represented as σ, is defined as the SEE susceptibility of an electronic device to ionizing radiation. Cross-section σ in radiation terms for proton and neutron interactions is the combination of sensitive area and probability of an interaction depositing the critical charge for a SEE. ***The SEU response of a device to a beam of particles is characterized as the SEU cross section, which is the number of upsets divided by the fluence of particles (p/cm2, particle flux integrated over the exposure time) to which the device was exposed.*** This data should be obtained by an experimental test in each component along a wide range of energy transferred to the device, because it highly depends on the internal design of bits. The units for cross-section are cm^2 per device or per bit."

25. Pag 38: Current solutions in airborne electronics.

"Single Error detection and Correction codes (SEC) as Hamming codes, [5], [6], [7], [28]. An error-correcting code (ECC) is a system that adds redundant bits to the original bitstream, being capable to recover even when a number of errors (up to the capability of the code being used). Error-correcting codes are usually distinguished between convolutional codes and block codes:

- ***SEC + Interleaving, [35], [36], [43]***

This technique traditionally selects the interleaving distance according to the maximum MCU size expected. There are several investigations that correlate the Interleaving selected against the error probability in order to minimize the implications for memory design impacting in area resources and timing delays. This technique, Interleaving, implies a routing that is more complex and a power consumption that is increased. Furthermore, a parity bit could be introduced for a SEC-DED for decoupling error detection from correction.

- ***Double Adjacent errors, SEC-DED-DAEC, [52], [59], [60], [63], [65]***

This technique is based on traditional SEC-DED codes (linear systematic block codes). Generator and Syndrome matrix are defined with the weight of all the data columns is three and the sum of any two adjacent data columns is four. This technique could correct the error pattern (--11--), where ones represents the errors affected, being physically close. SEC-DAEC codes are mistaken in double nonadjacent errors.

- ***Double Alternative Adjacent errors, SEC-DED-DAAEC, [63], [64]***

This technique is an extension of SEC-DED-DAEC. This technique could correct the error pattern (--101--), where bits affected are no physically close. Ones represents the errors affected and zeros are correct bits not affected by radiation event. SEC-DED-DAAEC codes are mistaken in double nonadjacent errors.

- ***Triple Adjacent errors, SEC-DAEC-TAEC, [63]***

This technique is an extension of SEC-DED-DAEC. This technique could correct the error pattern (--111--), where bits affected are physically close. Ones represent the errors affected. In that case, an additional parity check bit was required compared with a SEC-DED code. The decoding complexity is also larger than that for SEC-DED or SEC-DAEC codes as more syndromes have to be checked for error detection, therefore, it increases the cost of a high redundancy.

- ***SEC + Bloom filter [53], [54], [55], [62]***

A Bloom filter is a probabilistic data structure used to know if an element is a member of a set. False positive matches are possible, "possibly in set", but false negatives are not, "definitely not in set". As higher number of elements in the set the higher probability of false positives

- ***Built In-Current Sensors, [69]***

This technique proposes the use of sensors that measure the current, BICS, consumption in vertical lines of memory in order to detect the bits in which the error due to radiation event have been produced. This technique requires a coupled correction technique to clear the errors.

- ***DMR/TMR with approximate logics [70], [71]***

This technique proposes a duplication or triplication of system to be protected with modular approximation models of the principal functionality. This technique refers to functional similarity between logic functions or circuits instead of functional equivalence. Therefore, this technique is able to detect and correct errors within the principal functionality although its capability depends on the approximations used in the modular redundancies.

- ***Well-taps [61], [67]***

This technique proposes the use of well-taps distance (number of cells between the welltaps) due to its inhibiting physical effect on MCUs with an optimized interleaving distance to make parity codes viable for detecting and correcting errors. A higher number of well-taps, it increases the area,

- ***Temporary redundancy [68]***

This technique proposes the consecutive delayed transmission of a single data with multiple delayed clocks in order to detect temporary by means a majority voter the error in the transmission.

- ***Hsiao codes: [66]***

This technique is based on traditional linear systematic block codes. Generator and Syndrome matrix are defined with an odd-weight-column, no two columns are the same and there are no all-0 columns. This technique could correct the single error pattern.

- ***Orthogonal Latin Squares [56], [57]***

This technique is based on multiple error correction codes where the key parameters of an OLS code are its data block size k and the number of errors that can be corrected t. The block size is of the form k = m2 bits and to correct t errors, k = 2tm parity check bits are needed. The majority vote required to re-compute the four parity check bits and take a majority vote among them.”

26. Pag 43: Basic physical architecture

Extension added: “There are also some bits which are used for the scrubbing function itself (SEC1), which is identically duplicated (SEC2). ***Both scrubbers (SEC1 and SEC2) are implemented internally to the FPGA as part of the configuration FPGA bitstream. Therefore, the internal logic of both scrubbers is also susceptible to radiation.***

Traditionally, internal scrubbers are less reliable than external ones, because external scrubbers are placed into a radiation-hardened FPGA. This external implementation could achieve, in a first attempt, a lower failure probability due to technology immunity, although main drawbacks of this external scrubber are the complex HW architecture and cost increment.

The proposed Architecture allows having internal scrubbers and yet having fault tolerant performance. Failure probability calculations include possible faulty events at the internal scrubber logic”

27. Pag 105: Functional simulations

Extension added: “Fault-Tolerant HSB architecture development has been briefly presented in this Thesis. ***Several examples of different configurations of HSB Architecture have been simulated during architecture definition phase. Functional simulations have been performed in the definition phase in order to evaluate the performances and operation limits of HSB Architecture using Modelsim and Matlab.***

After the architecture definition phase, other simulation tools have been also used for error injection to check recovery capabilities of architecture. Errors have been introduced using Modelsim and SST (SEUs Simulation Tool by ESA and Nebrija University of Madrid).

Several examples of this architecture, Table 6.5, have been implemented and simulated in VHDL for different memory sizes. They have been synthetized and placed & routed into a Xilinx ***Spartan3E*** FPGA to check the feasibility of the system using the Xilinx ISE software. Results obtained depend quantitatively on the type of device in which the algorithm is implemented, although it enables having an example in terms of time and logic consumption for a qualitative assessment"

28. Pag 130: Timing analysis based on experimental results

Extension added: "Table 6.4 includes the ***operational logic timings for different technique implementations obtained after Synthesis and Place&Route processes performed with Xilinx ISE development tool. This analysis does not include the memory access timing. The analysis is performed*** for different memory sizes where different examples of timings are also included for different number of errors injected in the complete memory"

29. Pag 132: Resource analysis based on experimental results

Extension added: "Table 6.5 also includes the comparison of the results obtained from the Synthesis and Place&Route. ***Table 6.5 includes the operational logic resources for different technique implementations obtained after Synthesis and Place&Route processes performed with Xilinx ISE development tool. The implementation of memory, used for different EDAC comparison, has been developed based on FF implementation. Although BRAM allows a more efficient memory implementation with dedicated blocks of memory, the purpose of the qualitative comparison between techniques is the incremental difference between the results of each implementation.***

Furthermore, a comparison between HSB architecture and Hamming has been performed based on a BRAM memory implementation in order to validate the trends obtained with data shown in Table 6.5. The results of this comparison, shown in Figure 6.6., shown that BRAM memory required is the same for both techniques showing that the resources (LUTs and FFs) required for HSB architecture keeps the trends"

		8 Kbit	16 Kbit	32 Kbit	64 Kbit
HSB	BRAM	1 (272x32 bit)	1 (536x32 bit)	1 (1.048x32 bit)	1 (2.072x32 bit)
	LUTs	4,418	32,906	183,896	459,740
	FF	122	120	136	158
	Xor	42	46	50	56
	Adder 32 bit	4	4	4	4
	Counter 32 bit	2	2	2	2
HAM	BRAM	1 (272x32 bit)	1 (536x32 bit)	1 (1.048x32 bit)	1 (2.072x32 bit)
	LUTs	5,571	68,610	275,559	606,230
	FF	162	201	240	274
	Xor	42	48	54	60
	Adder 32 bit	6	6	6	6
	Counter 32 bit	3	3	3	3

Table 6.6: Comparison based on BRAM implementation

30. Pag 138: Summary of original contributions

Extension added: ***Meeting probability objectives (CAT & HAZ).***

According to probability analysis performed, there is a hierarchy of the factors involved when considering the failure rate, since each factor has a different order effect in it. This hierarchy is as follows:

1. ***The first failure rate level is the Single uncorrectable events, because the sum of Multiple Event probabilities is negligible against Uncorrectable single events. This implies that Interleaving distance shall be higher than MCU span in order to correct the error patterns generated by the single radiation events.***
2. ***Once Single events are corrected, the Double Uncorrectable events become the most important contributor to failure probability, because the sum of rest of Multiple event probabilities is negligible against Uncorrectable Double events***
3. ***Once Double events are corrected, the Triple Uncorrectable events become the most important contributor to failure probability, because the sum of rest of Multiple Event probabilities is negligible against Uncorrectable Triple events.***
4. ***The previous statements logic can be successively extrapolated.***

31. Pag 138: Summary of original contributions

Extension added: ***Architecture scalability for Fault-Tolerant systems***

The proposed Architecture allows a scalable design accordingly with the required failure rate in terms of the error hierarchy factor to be considered.

Traditional Fault-Tolerant architectures based on SEC-DED correction codes require a majority voting system to detect a failure in a scrubber functions. Therefore, these architectures are traditionally sized according to the required capability to correct simultaneous errors. For n errors impacted in same scrubbing cycle, 2·n+1 redundant scrubbers are required, Figure 7.1.

HSB architecture is not based on majority voting system. The capability to detect a failure in a scrubber functions is performed thanks to the first level of codification [73] and [74]. Therefore, this Fault-Tolerant architecture only requires n+ 1 redundancy for multiple radiations events. Thus, an area reduction for double and triple error correction capabilities are obtained compared with traditional systems based on majority voting systems.

	Number of redundancies required	
	SEC-DED	HSB
Single radiation event	3	2
Double radiation events	5	3
Triple radiation events	7	4
n radiation events	**2·n+1**	**n+1**

Figure 7.1. Scrubber redundancy comparison between HSB and Majority voting.

9.3 EXTRA REFERENCES

[43] S. Lee, S. Baeg, P. Reviriego, “Memory Reliability Model for Accumulated and Clustered Soft Errors” at IEEE Transactions on Nuclear Science, pp. 2483 – 2492, 2011.

[44] P. Reviriego, J. A. Maestro: “Optimizing Scrubbing Sequences for Advanced Computer Memories” at IEEE Transactions on Device and Materials Reliability, pp. 192 – 200, 2010.

[45] J. Maiz, S. Hareland, K. Zhang and P. Armstrong: “Characterization of Multi-bit Soft Error events in advanced SRAMs” at Technical Digest, IEEE International Electron Devices Meeting, 2003.

[46] S. Yoshimoto, T. Amashita, M. Yoshimura, Y. Matsunaga, H. Yasuura, S. Izumi, H. Kawaguchi, M. Yoshimoto: “Neutron-Induced Soft Error Rate Estimation for SRAM Using PHITS” at 2012 IEEE 18th International On-Line Testing Symposium (IOLTS)

[47] A. Jimenez, E. Novillo, F.Aguado: “Electromechanical actuator with anti-jamming system for safety critical aircraft applications” at International Congress Recent Advances in Avionics Actuation Systems and Components, 2014.

[48] A. Gallego: “Reliability and Safety Enhanced Electrical Actuation System Architectures” at International Congress of Aerodays, 2015.

[49] S. Ferreiro, A. Jimenez: “Prognostics and Health Monitoring for electro-mechanical Nose Landing Gear Door actuator of a UAV, based on simulation modelling and data-driven techniques” at International Congress of Prognostics and Health Management, 2013.

[50] A. Jimenez, E. Novillo, F.Aguado: “Load detection and fatigue Health Monitoring in Landing Gears” at International Mechanical Engineering Congress and Exposition, 2014

[51] J. D. Black, P. E. Dodd and K. M. Warren: “Physics of Multiple-Node Charge Collection and Impacts on Single-Event Characterization and Soft Error Rate Prediction” at IEEE Transactions on Nuclear Science, pp 1836 – 1851, 2013

[52] M. Richter, K. Oberlaenderz and M. Goessel: “New linear SEC-DED codes with reduced triple error miscorrection probability” at IEEE International On-Line Testing Symposium, pp 1836 – 1851, 2008

[53] S. Pontarelli and M. Ottavi: “Error Detection and Correction in Content Addressable Memories by Using Bloom Filters” at IEEE Transactions on Computers, vol. 62, no. 6, 2013

[54] M. Atamaner, O. Ergin, M. Ottavi, P. Reviriego, "Detecting errors in instructions with bloom filters", Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT) at IEEE International Symposium on, pp. 1-4, 2017.

[55] A. Sánchez-Macián, P. Reviriego, J.A. Maestro, S. Liu, "Single Event Transient Tolerant Bloom Filter Implementations", IEEE Transactions on Computers, vol. 66, no. 10, pp. 1831-1836, 2017.

[56] M. Demirci, P. Reviriego and J. A. Maestro: “Implementing Double Error Correction Orthogonal Latin Squares Codes in Xilinx FPGAs”, at Elsevier Ltd IN Microelectronics Reliability, Volume 56, Pages 221-227, 2016

[57] A. Sarkar, J. Samanta, A. Barman and J. Bhaumik: “FPGA Implementation of OLS (32, 16) Code and OLS (36, 20) Code” at Communication, Devices, and Computing. Lecture Notes in Electrical Engineering, vol 470. Springer, Singapore, 2017.

[58] P. Reviriego, J.A. Maestro, S. Baeg, S. Wen, and R. Wong: “Protection of Memories Suffering MCUs Through the Selection of the Optimal Interleaving Distance” at IEEE Transactions on Nuclear Science, vol. 57, no. 4, 2010

[59] P. Reviriego, J. Martínez, S. Pontarelli and J. A. Maestro: “A Method to Design SEC-DED-DAEC Codes With Optimized Decoding” at IEEE Transactions on device and materials reliability, vol. 14, no. 3, 2014

[60] A. T. Erozan, B. Tavli, "High performance adjacent error detection for nanometer devices", Electronics Letters, vol. 52, no. 21, pp. 1788-1789, 2016

[61] S. H. Jeon, S. Lee, S. Baeg, I. Kim, and G. Kim: “Novel Error Detection Scheme With the Harmonious Use of Parity Codes, Well-Taps, and Interleaving Distance” at IEEE Transactions on Nuclear Science, vol. 61, no. 5, 2014

[62] P. Reviriego, S. Pontarelli, J. A. Maestro, and M. Ottavi: “A Synergetic Use of Bloom Filters for Error Detection and Correction” at IEEE Transactions on Very Large Scale Integration (VLSI) systems, vol. 23, no. 3, 2015

[63] L. J. Saiz-Adalid, P. Reviriego, P. Gil, S. Pontarelli, and J. A. Maestro: “MCU Tolerance in SRAMs Through Low-Redundancy Triple Adjacent Error Correction” at IEEE Transactions on Very Large Scale Integration (VLSI) systems, vol. 23, no. 10, 2015

[64] P. Reviriego, S. Liu, L. Xiao, and J. A. Maestro: “An Efficient Single and Double-Adjacent Error Correcting Parallel Decoder for the (24,12) Extended Golay Code” at IEEE Transactions on Very Large Scale Integration (VLSI) systems, vol. 24, no. 4, 2016

[65] A.t Dutta and N. A. Touba: “Multiple Bit Upset Tolerant Memory Using a Selective Cycle Avoidance Based SEC-DED-DAEC Code” at IEEE VLSI Test Symmposium, 2007

[66] W. Wei, K. Namba, Y. Kim, and F. Lombardi: “A Novel Scheme for Tolerating Single Event/Multiple Bit Upsets (SEU/MBU) in Non-Volatile Memories” at IEEE Transactions on computers, vol. 65, no. 3, 2016

[67] K. Osada, K. Yamaguchi, Y. Saitoh, and T. Kawahara: “SRAM Immunity to Cosmic-Ray-Induced Multierrors Based on Analysis of an Induced Parasitic Bipolar Effect” at IEEE Journal of solid-state circuits, vol. 39, no. 5, 2004

[68] J. Gebelein: “Hamming FSM with Xilinx Blind Scrubbing” at Infrastructure and Computer Systems in Data Processing (IRI) Frankfurt University. 2012

[69] S. D. Dasnurkar and J. A. Abraham: “Calibration-Enabled Scalable Built-In Current Sensor Compatible with Very Low Cost ATE” at IEEE European Test Symposium, 2010

[70] A. J. Sanchez-Clemente, L. Entrena, R. Hrbacek, and L. Sekanina: “Error Mitigation Using Approximate Logic Circuits: A Comparison of Probabilistic and Evolutionary Approaches” at IEEE Transactions on Reliability, vol. 65, no. 4, 2016

[71] P. Balasubramanian and R. T. Naayagi: “Redundant Logic Insertion and Fault Tolerance Improvement in Combinational Circuits” at International Conference on Circuits, System and Simulation. Published by IEEE, 2017

[72] I. Herrera-Alzu and M. López-Vallejo: “Design Techniques for Xilinx Virtex FPGA Configuration Memory Scrubbers” at IEEE Transactions on Nuclear Science, vol. 60, no. 1, 2013

[73] A. Jimenez and J. Pleite: "Multiple Cell Upsets inside aircrafts. New Fault-Tolerant Architecture” at IEEE Transaction on Aerospace and Electronics System (JCR-Q1). IEEE 201700291, 2018

[74] A. Jimenez and J. Pleite: "Redundance technique for radiation protection in electronics for Airborne Safety Critical application”, Spanish patent: P201430854, 2015